Amiga Mail

Volume II

September/October 1990

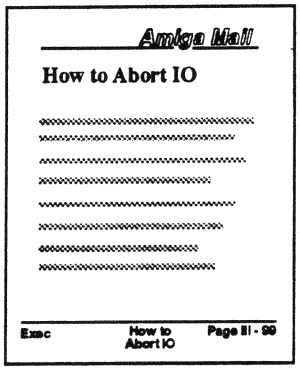
The Commodore Amiga Technical Newsletter

Copyright 1990 Commodore-Amiga, Inc. All rights reserved. Some of the materials used herein have been reproduced with the permission of the authors indicated. Use or reproduction of such materials shall be in accordance with the authors' instructions.

Amiga Mail
Volume II

Amiga Mail is published six times annually by the Commodore Applications and Technical Support Group (CATS). Its purpose is to support third-party developers of Amiga software and hardware world-wide by providing technical information and programming guidelines. Each issue features a collection of news and technical articles of special interest to Amiga product developers. The technical articles in each issue can be saved for future use in the Amiga Mail Tech Notes binder, a reference notebook containing all the articles from past issues of Amiga Mail.

For filing technical articles in the Amiga Mail Tech Notes binder, just use the chapter and page numbers listed in the footer of each article page. The Tech Notes chapter is shown as a Roman numeral followed by the page number within that chapter.



For instance, the article at right would be filed in chapter III of the Amiga Tech Notes binder, "Exec". The article starts on page 99 of that chapter in the binder.

For more information on the Amiga Mail Tech Notes binder and the Commodore Developer Support Program, contact your local support manager or if you are in the U.S., contact CATS at (215) 431-9180.

Warning

The information contained herein is subject to change without notice. Commodore specifically does not make any endorsement or representation with respect to the use, results, or performance of the information (including without limitation its capabilities, appropriateness, reliability, currentness or availability).

Disclaimer

This information is provided "as is" without warranty of any kind, either express or implied. The entire risk as to the use of this information is assumed by the user. In no event will Commodore or its affiliated companies be liable for any damages, direct, indirect, incidental, special or consequential, resulting from any defect in the information, even if advised of the possibility of such damages.

Amiga Mail News

EDC Set for Milan in February

The Fourth Annual European Developers Conference will be held in Milan, Italy from the 13th through the 16th of February at the Grand Hotel Brun. The deadline for registration is January 4th. Registration information along with a preliminary agenda will be sent to all registered developers in November.

Commodore ESCO has arranged for a hotel reservation service for the Grand Hotel Brun and an alternative hotel near by. The prices ESCO has negotiated are:

Grand Hotel Brun	Alternative hotel
180,000 Lira	160,000 Lira (Single)
250,000 Lira	210,000 Lira (Double)

Prices include breakfast and local value added tax.

Although the details haven't been finalized, the conference sessions are scheduled for the 13th through the 15th. The 16th has been set aside as "Show-off and Developers Forum Day". This day gives developers a chance to meet with Commodore and with each other to discuss their latest software and hardware projects and ideas.

ESCO is also considering video taping some of the more important sessions, giving developers the opportunity to see sessions that are scheduled for the same time. The application will include a ballot for attendees to vote on the sessions they most want to attend.

Welcome to Volume II

This issue marks the beginning of Amiga Mail's second volume. Amiga Mail is now dedicated to covering release 2 of the Amiga OS as well as the rest of Commodore's new Amiga hardware and software. Some of the new topics will include ARexx, networking, CDTV, and multimedia.

The new volume has also gone through some minor cosmetic changes as well, the most noticeable of which is in the format of code listings. Pages of code listings will be formatted as dual-columned landscape pages, much like the Includes and Autodocs appear in the *Includes and Autodocs Manual*. This method will compact lengthy source code listings, leaving room for more articles.

As with the first volume, a binder and tabs are available from CATS. The binder and tabs provide an easy way to save the articles for later reference. Certified developers and non-developers can purchase a new binder and tabs through the normal CATS channels. The price of the binder and tabs is $10 and $5, respectively.

IN THIS ISSUE

TagItems and Tag Lists
Staff

Obtaining a Disk Object
David Junod

An Introduction to V36 Screens and Windows
Ewout Walraven

Master Index for Amiga Documentation
Staff

Virginia Tech Chooses Commodore for Unix Platform

Commodore Business Machines, Inc. scored a "pre-season" victory with an order for Unix-based Amiga 3000s by Virginia Tech. The Unix based Amiga has not been formally announced as a product.

Since 1984, the university has required their computer science majors to purchase a Unix-based computer system. At first the students had to buy Apple's Lisa, which ran a version of Unix from Unipress, but later, in 1987, the school switched to Apple's A/UX running on a 68020 based Macintosh. After examining Unix implementations on a variety of platforms, Virginia Tech decided to abandon the Apple configuration for Commodore's.

"This is a significant win for Commodore, because it clearly demonstrates that power and price-performance superiority of the Amiga 3000," said Howard Diamond, Commodore's Director of Education. "The marriage of the Amiga and V.4 technology is an exciting development for us and for our customers in higher education and other markets," he added.

Commodore's Unix is an implementation of AT&T's System V release 4. The design goal of System V release 4 (SVr4) is to integrate the features of the various existing Unix implementations into a unified standard. Over 150 companies have declared their support for SVr4. Unix technology, in general, is significant because it establishes common standards for the exchange of information between dissimilar hardware systems, an especially critical need for computer science and other university departments.

"Commodore's Unix passed every one of our tests," said Dr. James Arthur, chairman of the computer science department's selection committee. "From the Amiga technology to V.4 delivery to the Amiga's unbeatable price performance, Amiga Unix was the clear choice for our department. We're also very pleased with Commodore's willingness to work closely with us to produce a solution tailored to our needs." Virginia Tech selected Unix as their operating system standard because it provides an ideal environment for experimentation and because it is largely platform-independent.

Virginia Tech's purchase comes during the product's beta test phase. At press time, Amiga Unix platforms were not generally available.

Over the years, technology advances have prompted Virginia Tech's constant re-evaluation of the required platform. The selection of the Commodore Amiga is a realization of that re-evaluation process. According to Dr. Arthur, Virginia Tech is the only university in the country which requires its computer science majors to purchase a particular Unix platform. Computer science students become intimately familiar with the Unix operating system and underlying concepts found on a wide range of computers from mainframes to micros, and will use the Amiga to perform homework, in-class projects and term papers.

TagItems and Tag Lists

Staff

The implementation of tags is one of the many new features of release 2.0. Tags made it possible to add new parameters to system functions without interfering with the original parameters. They also make specifying parameter lists much clearer and easier.

A tag is made up of an attribute/value pair as defined below (from *utility/tagitem.h*):

```
struct TagItem
 {
     ULONG  ti_Tag;    /* identifies the type of this item */
     ULONG  ti_Data;   /* type-specific data, can be a pointer */
 };
```

The ti_Tag field specifies an attribute to set. The possible values of ti_Tag are implementation specific. System tags are defined in the include files. The value the attribute is set to is specified in ti_Data. An example of the attribute/value pair that will specify a window's name is:

```
ti_Tag  = WA_Title;
ti_Data = "My Window's Name";
```

The ti_Data field often contains 32-bit data as well as pointers.

One way tags are passed to system functions is in the form of *tag lists*. A tag list is an array or chain of arrays of TagItem structures. Within this array, different data items are identified by the value of ti_Tag. Items specific to a subsystem (Intuition, Graphics,...) have a ti_Tag value which has the TAG_USER bit set. Global system tags have a ti_Tag value with TAG_USER bit clear. The global system tabs include:

```
TAG_IGNORE - A no-op.  The data item is ignored.
TAG_MORE   - The ti_Data points to another tag list, to support chaining of TagItem arrays.
TAG_DONE   - Terminates the tag item array (or chain).
TAG_SKIP   - Ignore the current tag item, and skip the next n array elements, where
                n is kept in ti_Data.
```

Note that user tags need only be unique within the particular context of their use. For example, the attribute tags defined for OpenWindow() have the same numeric value as some tags used by OpenScreen(), but the same numeric value has different meaning in the different contexts.

System functions receive TagItems is several ways. One way is illustrated in the Intuition function OpenWindow(). This function supports an extented NewWindow structure called ExtNewWindow. When the NW_EXTENDED flag is set in the ExtNewWindow.Flags field, OpenWindow() assumes that the struct TagItem *Extension field contains a pointer to a tag list.

Another method of passing a tag list is to directly pass a pointer to a tag list, as OpenWindowTagList() does in the following code fragment.

```
struct TagItem tagitem[3];
struct Window *window;

tagitem[0].ti_Tag = WA_CustomScreen;
tagitem[0].ti_Data = screen;     /* Open on my own screen */
tagitem[1].ti_Tag = WA_Title;
tagitem[1].ti_Data = "AmigaMail Test Window";
tagitem[2].ti_Tag = TAG_DONE;    /* Marks the end of the tag array. */

/* Use defaults for everything else. Will open as big as the screen. */

if (window = OpenWindowTagList(NULL, /* Because all window parameters
                                      * are specified using tags, we don't
                                      * need a NewScreen structure */
                               tagitem))
{
    /* rest of code */
    CloseWindow(window);
}
```

Notice that window parameters need not be explicitly specified. Functions that utilize tags should have reasonable defaults to fall back on in case no valid attribute/value pair was supplied for a particular parameter. This fall back capability is a useful feature. An application only has to specify the attributes that differ from the default, rather than unnecessarily listing all the possible attributes.

The *amiga.lib* support library offers an example of another way to pass TagItems to a function. Rather than passing a tag list, the function OpenScreenTags() receives the attribute/value pairs in the argument list, much like printf() receives its arguments. Any number of attribute/value pairs can be specified. The following code fragment illustrates the usage of OpenScreenTags().

```
struct Screen *screen;

screen = OpenScreenTags(NULL, /* Don't bother with a NewScreen structure */
                        SA_Overscan, OSCAN_TEXT,
                        SA_Title, "Amiga Mail Screen",
                        TAG_DONE );
```

Tags are not exclusively for use with the operating system; the programmer can implement them as well. The run-time library *utility.library* contains several functions to make using tags easier. For more information on *utility.library*, see the appropriate release 2.0 Autodocs.

Amiga Mail
Volume II
September/October 1990

Obtaining an Icon Image

By David N. Junod

One of the many improvements made to Workbench for release 2.0 lies in its ability to supply an application with arguments from Workbench. Prior to release 2.0, the only Workbench supported way an application could receive Workbench arguments was through the Workbench startup message. Now, Workbench arguments can come from many sources including AppWindows, AppIcons, and the ASL file requester. These enhancements have made processing Workbench arguments more beneficial than ever.

When an application is started via Workbench, the application receives a startup message (struct WBStartup) from Workbench. This message contains a pointer to an array of WBArg structures. The array of WBArgs corresponds to all Workbench icons that were selected at the time the application was launched. Each WBArg structure contains a file lock on the file or directory associated with each icon, as well as the name associated with the icon. A WBArg supplies the link to the icon, or the DiskObject, itself, from which necessary information, such as application or project ToolTypes, can be extracted.

Under release 2.0, Workbench supplies WBArgs using several different structures. The structure used depends on the source of the WBArgs. When it launches an application, Workbench sends the application WBArgs through the WBStartup structure. AppWindows and AppIcons use the AppMessage structure to send WBArgs. The ASL file requester provides the arguments in the FileRequester structure.

Also with release 2.0, it is possible that icons passed to an application do not have an *.info* file associated with them. Workbench can now display *all* files and directories iconically, not just those with *.info* files. For icons without *.info* files, Workbench supplies a default icon based on the protection bits and object type (file, dir, etc.) so the user can manipulate files and directories from Workbench.

The following routine uses GetDiskObjectNew() which should be used only when an application requires an image for every Workbench argument, whether actual or default. To only read actual icons and tooltypes, modify the routine to use GetDiskObject() instead. For more information, see the release 2.0 Autodocs.

```
/* wbarg.c
 * How to get an icon from a Workbench argument
 * Written by David N. Junod, 27-Aug-90
 */

#include <exec/types.h>
#include <exec/memory.h>
#include <exec/libraries.h>
#include <libraries/dos.h>
#include <workbench/icon.h>
#include <workbench/startup.h>
#include <clib/dos_protos.h>
#include <clib/exec_protos.h>
#include <clib/wb_protos.h>
#include <pragmas/dos.h>
#include <pragmas/exec.h>
#include <pragmas/wb.h>
#include <string.h>

extern struct Library *IconBase, *DOSBase;

struct DiskObject *GetDiskObjectNew (STRPTR name);
struct DiskObject *GetDiskObject (STRPTR name);

#pragma libcall IconBase GetDiskObjectNew 84 801

/****** wbarg/IconFromWBArg ******************************************
*
*   NAME
*      IconFromWBArg - Obtains the icon from a Workbench argument.
*
*   SYNOPSIS
*      dob = IconFromWBArg (wbarg);
*
*      struct DiskObject *dob;
*      struct WBArg *wbarg;
*
*   FUNCTION
*      Given a valid Workbench argument, this command will return the
*      proper icon.  This call uses the new (V36) GetDiskObjectNew call
*      to ensure that an icon will be returned.
*
*      This functions requires dos.library and icon.library to be opened
*      by the application.
*
*   INPUTS
*      wbarg  - Pointer to a filled in WBArg structure.
*
*   RESULTS
*      dob    - Pointer to a DiskObject structure.  Application must call
*               FreeDiskObject when done with the icon.
*
*********************************************************************/

struct DiskObject *IconFromWBArg (struct WBArg * arg)
{
    struct DiskObject *dob = NULL;
    UBYTE work_name[34];
    BPTR old, new;

    /* Copy the WBArg contents */
    strcpy (work_name, arg->wa_Name);

    /* Duplicate the lock */
    if (new = DupLock (arg->wa_Lock))
    {
       BOOL success = FALSE;

       /*
        * If we don't have a name, then get one. Only Tools & Projects have a
        * valid name.
        */
       if (strlen (work_name) == 0)
       {
           LONG msize = sizeof (struct FileInfoBlock);
           struct FileInfoBlock *info;

           /* This block needs to be longword aligned, so allocate it */
           if (info = (struct FileInfoBlock *) AllocMem (msize, MEMF_CLEAR))
           {
              /* Examine the lock, so that we can figure out the name */
              if (Examine (new, info))
              {
                  /* Copy the name of the lock */
                  strcpy (work_name, info->fib_FileName);

                  /* See if the lock is a directory */
                  if (info->fib_DirEntryType > 0)
                  {
                     /* Unlock the existing lock */
                     UnLock (new);

                     /* Go to the parent drawer */
                     if ((new = ParentDir (arg->wa_Lock)) == NULL)
                     {
                         /* A disk icon was passed */
                         strcpy (work_name, "disk");
                         new = DupLock (arg->wa_Lock);

                     }       /* End of if disk */
                  }          /* End of if directory */

                  /* We have a name */
                  success = TRUE;

              }              /* End of examine */

              /* Free the temporary block */
              FreeMem ((APTR) info, msize);
           }
       }
       else
       {
           /* Show that we have a name to work with */
           success = TRUE;
       }

       /* See if we should try getting the icon */
       if (success)
       {
           /* go to the directory where the icon resides */
           old = CurrentDir (new);

           /*
            * GetDiskObjectNew is a new function, for AmigaOS version 2.0,
            * therefore we need to check the version level.
            */
           if (IconBase->lib_Version >= 36)
              dob = GetDiskObjectNew (work_name);
           else
              dob = GetDiskObject (work_name);

           /* go back to where we used to be */
           CurrentDir (old);
       }

       /* release the duplicated lock */
       UnLock (new);
    }

    /* return the icon */
    return (dob);
}
```

❖

An Introduction to V36 Screens and Windows

by Ewout Walraven

Editor's Note: The following article assumes the reader is familiar with the concept and implementation of tags in Intuition under release 2.0. More information on the subject can be found in the Amiga Mail article "TagItems and Tag Lists".

The new *intuition.library* provides the programmer with flexible yet simple methods of opening Intuition screens and windows. This article is an introduction to some of the Intuition and relevant *graphics.library* functions that make this possible. For more in depth information see the 1990 DevCon notes as well as the Autodoc and include files for release 2.0.

The Display Database

Release 2.0 of the OS introduces system support for different monitor types and programmable display modes. Applications can make use of these features through the *Display Database*. The Display Database provides an easy way for applications to stay compatible with past, present and future versions of the OS and the display hardware.

The database contains information about the available monitor types (like PAL or VGA) and the display modes that a particular monitor type is capable of displaying (like Lores, Hires, SuperHires, etc). This database makes it possible for the user to dynamically add new monitor types to the system, typically by calling the *addmonitor* command using a monitor spec file as an argument. These monitor types will be added to the database after the default monitor. The default monitor properties are determined at boot time based on the configuration of the custom chip set. The default monitor is always either an alias for the NTSC or PAL monitor type.

When a monitor type is added to the database, the system figures out which display modes are available to this monitor type, based on the properties of the monitor's spec file and the configuration of the custom chip set. Later, the Display Database can be queried on the availability of a particular display mode using the appropriate *graphics.library* calls.

The following is an example of how to use *graphics.library* functions to find out what display modes are available. See the 2.0 Autodocs for a detailed description of the various functions.

```
ULONG modeID;
ULONG skipID;

ULONG error, result;
struct DisplayInfoHandle displayhandle;
struct DisplayInfo displayinfo;
struct NameInfo nameinfo;

/* graphics.library must be opened */

/* Don't want duplicate entries in the list for the
 * 'default monitor', so we'll skip the the videomode
 * for which default.monitor is the alias.
 */

/* INVALID_ID indicates both the start and the end of the list of
   available keys */
modeID = INVALID_ID;

GetDisplayInfoData(NULL, (UBYTE *) & displayinfo,
        sizeof(struct DisplayInfo), DTAG_DISP, LORES_KEY);
if (displayinfo.PropertyFlags & DIPF_IS_PAL)
    skipID = PAL_MONITOR_ID;
else
    skipID = NTSC_MONITOR_ID;


/* Given a ModeID, NextDisplayInfo
                returns the next ModeID in the list */
while((modeID = NextDisplayInfo(modeID)) != INVALID_ID) {

    /* Skip it? */
    if (modeID & MONITOR_ID_MASK != skipID) {

        /* ModeNotAvailable returns NULL if a displaymode,
           specified by the modeID, is available, or an error
           indicating why it is not available.
        */
        if ((error = ModeNotAvailable(modeID)) == NULL) {

            /* This displaymode is available, get the naming
               information. GetDisplayInfoData, can either be
               called with a handle to a displaymode record, or
               the display modeID. Never use the handle directly.
            */

            /* returns NULL if not found */
            if (displayhandle = FindDisplayInfo(modeID)) {
                result = GetDisplayInfoData(displayhandle,
                        (UBYTE *)&nameinfo, sizeof(struct NameInfo),
                        DTAG_NAME, NULL);
                if (result)     /* 'result' indicates the number
                                   of bytes placed in the buffer */
                    printf("%s is available.\n", nameinfo.Name);
            }
        }
    }
}
```

Using the GetDisplayInfoData() function, this example asks only for the descriptive name of the display mode via the NameInfo structure. Using this same function with a different tag, other information on a particular display mode can be obtained. GetDisplayInfoData() can supply information about the properties of the display mode (DisplayInfo), the display mode's dimensions (DimensionInfo) or the display mode's monitor specifications (MonitorInfo). Refer to *graphics/displayinfo.h* for a description of the various structures.

Opening Screens and Windows

Intuition V36 supplies two new functions to open screens and windows: OpenScreenTagList() and OpenWindowTagList(). They provide added functionality over the older OpenScreen() and OpenWindow() functions. Instead of the NewScreen and NewWindow structures, an array of parameters called *tags* is passed to the function in the form of a TagItem structure pointer. This array is known as a *tag list*. For a description of using tags in Intuition, see the Amiga Mail article "TagItems and Tag Lists".

The following is a simple example of the usage of OpenScreenTagList() and OpenWindowTagList(), using only a tag list as a parameter to each function.

```
struct TagItem tagitem[3];
struct Screen *screen;
struct Window *window;

tagitem[0].ti_Tag  = SA_DisplayID;
tagitem[0].ti_Data = HIRES_KEY;      /* Use HIRES displaymode
                                        for default monitor */
tagitem[1].ti_Tag  = SA_Title;
tagitem[1].ti_Data = "AmigaMail Test Screen";
tagitem[2].ti_Tag  = TAG_DONE;       /* Marks the end of the tag array. */

/* All others options will be set to default. Note this screen will not
   open with the 'New Look', due to the absence of the SA_Pens tag and
   data. Further examples will have this tag */

if (screen = OpenScreenTagList(NULL,   tagitem)) {
    tagitem[0].ti_Tag = WA_CustomScreen;
    tagitem[0].ti_Data = screen;     /* Open on my own screen */
    tagitem[1].ti_Tag = WA_Title;
    tagitem[1].ti_Data = "AmigaMail Test Window";
    tagitem[2].ti_Tag = TAG_DONE;    /* Marks the end of the tag array. */

    /* Use defaults for everything else. Will open as big as the screen. */

    if (window = OpenWindowTagList(NULL, tagitem)) {

        /* rest of code */

        CloseWindow(window);
    }
    CloseScreen(screen)
}
```

See the *intuition/intuition.h* and *intuition/screens.h* include files for a complete list of currently supported tags.

Intuition.library provides a function to get information about screens and display modes called GetScreenDrawInfo(). This function returns a subset of the data supplied by the *graphics.library*'s GetDisplayInfoData() function. Given a pointer to a screen, GetScreenDrawInfo() returns a pointer to a DrawInfo structure describing that screen. Note that this structure is ***read-only***. The DrawInfo structure contains information about the resolution, default font and pen colors of the screen. Since the system allocates the DrawInfo structure when GetScreenDrawInfo() is called, a program must tell the system to free it by calling FreeScreenDrawInfo(). Of course, a program should only release the DrawInfo structure after it is no longer needed.

When using GetScreenDrawInfo() on a public screen, the screen should first be locked using Intuition's LockPubScreen() function. This will prevent a public screen from closing while your program is looking at the screen's DrawInfo structure. UnlockPubScreen() will unlock the screen. Public screens are discussed in more detail later in this article.

The *graphics.library* function GetVPModeID() provides a way of determining the display mode ID associated with a ViewPort. Passed the ViewPort attached to a screen, the function will return the display mode ID for that ViewPort, or `INVALID_ID` if no display mode ID is associated with the ViewPort.

An example of the usage of GetVPModeID() and the other functions discussed so far in this article are provided in at the end of thisarticle in example 1, *CloneWorkbench.c*. This example also uses the stack-based *amiga.lib* support library functions OpenScreenTags() and OpenWindowTags(), both of which are documented in the release 2.0 Autodocs.

Public Screens

V36 introduces the *public* screen, a custom screen on which other applications can open their windows. A familiar example of a public screen is the Workbench screen. The public screen provides the ability to create system supported environment alternatives to the Workbench.

Opening a public screen is just like opening a custom screen. The only difference is in the tags used when the screen is opened. There must be a `SA_PubName` tag, with the screen's name as the tag's data. Applications will access the screen by this name. When a public screen is first opened, it is still private to the application that opened it. This will give the program that opened the screen the opportunity to set up before other applications can access the screen.

After a public screen is explicitly made public, anybody can put a lock on the screen or open a window on it. Both locks and 'visitor windows' prevent applications from closing their public screens. Intuition keeps track of the number of locks and visitor windows on a public screen by incrementing and decrementing the screen's visitor count. When the number of visitors is zero, the screen can be closed.

The following code fragment shows how to open a public screen and change its status from private to public using the Intuition function PubScreenStatus().

```
if (pubscreen = OpenScreenTags(NULL, SA_DisplayID, HIRES_KEY,
                    SA_Depth, 2, SA_Title, "AmigaMail Test Public Screen",
                    SA_PubName, "AmIgaMail Test Public Screen",
                    /* By providing the public name, you indicate
                       it will be public */
                    SA_Pens, dri_Pens,
                       SA_ErrorCode, &oserror,
                       TAG_END)) {
    /* change mode from 'private' to 'public' */
    oldstatus = PubScreenStatus(pubscreen, 0);}
```

If for any reason Intuition could not open the screen, the error number will be returned in the data portion of the `SA_ErrorCode` tag (the OpenScreen errors are listed in *intuition/screens.h*). This error can indicate that the system is out of memory, that the requested display mode is not supported by the video hardware or monitor, or that the public screen name is already being used by another public screen.

Some OpenScreen errors can be avoided. Make sure the display mode a screen requires is available (one of the previous examples shows how to find all available display modes). For example, before opening a public screen, make sure its name is not already being used by an existing public screen. Before opening a public screen named *ExamplePublicScreen*, try to put a lock on a screen with that name using LockPubScreen(). If this call succeeds, the system has found and locked a screen with that name, so you can't use it. If the call fails, the system does not have a screen by that name, indicating that it's OK to open the new screen with that name. An application should make sure it removes all locks on a screen when the screen is no longer needed.

If a screen already exists with that name, there are several options available, all of which should reflect user preferences. One option is not to open a screen, using the existing screen instead. For example, if the user double-clicks an application's project icon, the application would check if its public screen is already opened and open windows there before trying to open a new screen. Another option is to open a new screen with a slightly different name. A public screen that would have been named *ExamplePublicScreen* might instead be called *ExamplePublicScreen2*. A third option is to quit. In any case, the user should normally be choosing the actions of the application, not the programmer.

Programs find specific public screens by their name, so public screens must have a unique name as identification. With a maximum name length of 139 characters, supplying a unique name shouldn't be a problem. This name should reflect the nature of the screen's purpose, if for no other reason, to make it easy to figure out what the screen is used for. Ambiguous names such as "PublicScreen" or "MyScreen" should not be used as they tell nothing about the nature of the screen. Also, more than one "less creative" programmer might be inclined to give such generic names to their public screen, which can confuse both user and application.

As a public screen can be used as an alternative to the Workbench environment, programs opening public screens should normally make the screen is at least 640 pixels wide. Many applications assume that the Workbench is at least 640x200 and would be severely crippled if opened on a 320x200 screen.

Before a public screen can be closed, it has to be made private again. As long as there are visitors (windows and locks) on a public screen, an attempt to change the screen's status to private will fail. To make closing down public screens easier, Intuition can signal an application when the last visitor 'leaves' a public screen. While waiting to close a locked public screen, an application can start to shut down by unloading code or closing windows. After the screen has been made private again, Intuition removes it from the public screen list. The following code illustrates this.

```
if ((oldstatus = PubScreenStatus(pubscreen, PSNF_PRIVATE)) & PSNF_PRIVATE)
        CloseScreen(pubscreen);
```

To tell Intuition to signal when all the visitor windows on a public screen have been closed, use the SA_PubSig and SA_PubTask tags when opening the screen. The SA_PubSig data is the number of a signal bit you have allocated. An application can specify which task to signal when all the visitors have left by supplying a pointer to that task with the SA_PubTask tag. If this tag is absent or its data = NULL, the task which opened the public screen will be signaled.

When putting up any requesters on a public screen, be sure to supply the requester with a pointer to a visitor window on that screen. This will prevent the requester from popping up on the wrong screen. This will also prevent needless incrementing of the public screen's visitor count as the requester will not be considered a visitor.

Prior to release 2.02, UnlockPublicScreen() did not signal when the visitor count reached zero. With release 2.02, UnlockPubScreen() will signal the specified task when the visitor count reaches zero. Under 2.00 and 2.01, only CloseWindow() would signal.

There is always a default public screen available. Normally this would be the Workbench, but an application can make its own public screen the default, or better yet, a screen manager utility could let the user choose the default public screen. An application should not automatically make its public screen the default as this might confuse the user and can conflict with user preferences.

Because older programs do not know about public screens, The SHANGHAI flag has been provided to 'shanghai' requests to open a window on the Workbench screen, forcing the window to open on the default public screen. Of course, a release 2.0 application that wants to open a window on a specific public screen can do so provided that the specific screen is available. Another flag, POPPUBSCREEN, instructs Intuition to move public screens to the front if a window is opened on it. The SHANGHAI and POPPUBSCREEN flags can be set using PubScreenStatus() but, as the flags are global and should be set according to the preferences of the user, applications should not be changing them.

Example 2, Pub.c, shows how to open a public screen, make it the default and shut it down. The example can be found at the end of this article.

To aid 'screen manager' utilities, the list of public screens can be locked and copied using LockPubScreenList(). UnlockPubScreenList() will unlock the list. Don't try to interpret the list itself. Instead, copy it and release the lock. To receive an updated copy later on, lock and copy the list again. The following is a small example of locking the public screen list.

```
/*
 * V36 Lock public screen list example
 *
 * Compiled with SAS C 5.10:  lc -cfist -v -L locklist.c
 * Use V36 include files.
 *
 */

#include <string.h>

#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <exec/memory.h>
#include <exec/lists.h>

#include <clib/intuition_protos.h>
#include <clib/exec_protos.h>
#include <clib/alib_protos.h>

void main (void);

struct IntuitionBase *IntuitionBase;

void main(void)
{

    struct List *publist;
    struct List *copy_publist;
    struct PubScreenNode *psnode;
    struct PubScreenNode *copy_psnode;

    /* fails silently if not V36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36)) {
        if (copy_publist = AllocMem(sizeof(struct List), MEMF_CLEAR)) {
            NewList(copy_publist);

            /* Lock the public screen list */
            publist = LockPubScreenList();

            /* and copy it */
            for (psnode = (struct PubScreenNode *)publist->lh_Head;
                 psnode->psn_Node.ln_Succ;
                 psnode = (struct PubScreenNode *)psnode->psn_Node.ln_Succ) {
                if (copy_psnode = AllocMem(sizeof(struct PubScreenNode), MEMF_CLEAR)) {
                    /* Copy the structure */
                    *copy_psnode = *psnode;

                    /* ln_Name points to the public screen name, make your own copy */
                    if (copy_psnode->psn_Node.ln_Name =
                            AllocMem(strlen(psnode->psn_Node.ln_Name) + 1, MEMF_CLEAR))
                        strcpy(copy_psnode->psn_Node.ln_Name, psnode->psn_Node.ln_Name);
                    else
                        printf("Not enough memory to copy screen name\n");

                    AddTail(copy_publist, (struct Node *)copy_psnode);
                } else {
                    printf("out of memory\n");
                    break;
                }
            }
            UnlockPubScreenList();

            psnode = (struct PubScreenNode *)copy_publist->lh_Head;
            while (copy_psnode = (struct PubScreenNode *)psnode->psn_Node.ln_Succ) {
                printf("%s\n", psnode->psn_Node.ln_Name);

                printf("psn_Screen       0x%lx\n", psnode->psn_Screen);
                                                    /* a pointer to the public screen */
                printf("psn_Flags        0x%lx\n", psnode->psn_Flags);
                                                    /* the flags */
                printf("psn_VisitorCount %ld\n", psnode->psn_VisitorCount);
                                                    /* the number of visitorwindows */
                printf("psn_SigTask      0x%lx\n", psnode->psn_SigTask);
                                                    /* The task to signal when the */
                                                    /* visitor is gone */
                printf("psn_SigBit       %ld\n\n", psnode->psn_SigBit);
                                                    /* The signalbit number to use */
                                                    /* to signal the task */

                if (psnode->psn_Node.ln_Name)
                    FreeMem(psnode->psn_Node.ln_Name, strlen(psnode->psn_Node.ln_Name) + 1);
                Remove((struct Node *)psnode);
                FreeMem(psnode, sizeof(struct PubScreenNode));
                psnode = copy_psnode;
            }
            FreeMem(copy_publist, sizeof(struct List));
        }
        CloseLibrary(IntuitionBase);
    }
}
```

Visitor Windows

Opening a visitor window on a public screen is like opening a window on a custom screen except that the window is specifically defined as a visitor. This is done when opening the window by either replacing the CUSTOMSCREEN flag in the (Ext)NewWindow->Type field with PUBLICSCREEN or using a WA_PubScreenName or WA_PubScreen tag. To use the PUBLICSCREEN flag or the WA_PubScreen tag (this tag has a pointer to the public screen as data), the screen must be prevented from closing. Normally this is done by locking the desired public screen. In either case the program must provide a pointer to the public screen if it requires a specific public screen. If an application has no preference towards a particular screen, it can pass NULL as the data for the WA_PubScreen tag to open a window on the default public screen.

The following example shows how to open a visitor window on a specific public screen or, if that screen can't be found, on the default public screen.

```
struct Screen *screen;
struct Window *window;
UBYTE namebuffer[MAXPUBSCREENNAME];

if (!(screen = LockPubScreen("CATScreen"))) {
    /* Can't lock CATScreen, fall back on default */
    /* The method used in this example to fall back on the default
       screen is not necessary and is provided only to illustrate
       the usage of GetDefaultPubScreen().
    */
    GetDefaultPubScreen(namebuffer);
    /* Lock it, so it can't go away. */
    /* If in this split second another screen is made the default
       and this one closes, screen would become NULL and the window
       will open on the default public screen anyway.
    */
    screen = LockPubScreen(namebuffer);
}
if (window = OpenWindowTags(NULL,
                WA_Left, 0, WA_Width, screen->Width,
                WA_Top, screen->BarHeight,
                WA_Height, screen->Height - screen->BarHeight,
                WA_PubScreen, screen,
                WA_Flags,
                WINDOWDRAG|WINDOWDEPTH|WINDOWCLOSE|ACTIVATE|
                        SIMPLE_REFRESH|NOCAREREFRESH,
                WA_Title, "AmigaMail Visitor window", TAG_END)) {
    /* Don't need the lock anymore. */
    UnlockPubScreen(NULL, screen);
                    .
                    .
                    .
    CloseWindow(window);
} else UnlockPubScreen(NULL, screen);   /* OpenWindow failed for
                                           some other reason */
```

In this code example, once it is locked (first by LockPubScreen(), later by the visitor window) the screen can't go away until all its visitors have left. After opening its initial visitor window, an application can open non-visitor windows on a public screen, however these windows will have to be closed *before* releasing the lock on the public screen.

When opening a visitor window, the public screen on which it opens can be specified by using a pointer to the screen (with the WA_PubScreen tag) or by supplying the screen name (using the WA_PubScreenName tag). This latter method is particularly useful where an application prefers a specific public screen, but can run using the default public screen. If the preferred public screen is not present when an application attempts to open a visitor window, the window will either open on the default public screen or will fail to open depending on the WA_PubScreenFallBack tag. If this tag is present and is set to TRUE, the window will open on the default screen. If WA_PubScreenFallBack is not present or is set to FALSE, the window will not open.

Overscan

In release 2.0, Intuition fully supports video overscan. There are four standard overscan specifiers: OSCAN_TEXT, OSCAN_STANDARD, OSCAN_MAX and OSCAN_VIDEO. OSCAN_TEXT is based on user preference settings and indicates a display which is within the visible bounds of the monitor. The dimensions of this type of overscan are used for the Workbench screen and for the defaults for OpenScreenTagList(), STDSCREENWIDTH and STDSCREENHEIGHT. OVERSCAN_STANDARD is also based on user preference settings. The edges of an OVERSCAN_STANDARD display are set by the user via *Preferences* to be just outside the visible bounds of the monitor. OSCAN_MAX is the largest display fully supported by Intuition. OSCAN_VIDEO is the largest display region which, for a given display mode, can be reliably generated. Note that OSCAN_VIDEO displays are not scrollable.

The Intuition function QueryOverscan() returns rectangle information for the indicated overscan type of a particular display mode. Using the rectangle returned by QueryOverscan(), it is possible to explicitly specify the offset and dimensions of the DisplayClip region. Note however that this custom rectangle must fit within the OSCAN_MAX rectangle, offset included. It is not permitted to specify custom rectangles whose values are in between OSCAN_MAX and OSCAN_VIDEO, nor is it permitted to specify rectangles larger than OSCAN_VIDEO.

DisplayClip

The DisplayClip defines the on-monitor region, and is independent of the screen dimension. The rectangle describing the DisplayClip determines how much of the screen can be seen at a time. The screen is "shown through" the DisplayClip in much the same way a SuperBitmap is shown through a window. If the screen is larger than the DisplayClip, only part of the screen will be visible. The DisplayClip makes it is possible for Intuition to support hardware scrolling of a raster larger than can fit on the display. Intuition will take care of specifying the rectangle of the DisplayClip region based on the default screen dimensions or based on the SA_Overscan tag (if it is present), so most programs shouldn't have to specify the DisplayClip rectangle.

Example 3, *ScreenDisplayModes.c*, illustrates opening any type of screen currently available to the system, using any type of available overscan.

```
/*
 * V36 Simple public screen example.
 *
 * Compiled with SAS C 5.10:  lc -cfist -v -L pub.c
 * Use V36 include files.
 */

#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <graphics/displayinfo.h>
#include <libraries/gadtools.h>
#include <exec/memory.h>
#include <dos/dosextens.h>

#include <clib/exec_protos.h>
#include <clib/gadtools_protos.h>
#include <clib/intuition_protos.h>

#include <pragmas/exec_pragmas.h>
#include <pragmas/gadtools_pragmas.h>
#include <pragmas/intuition_pragmas.h>

struct IntuitionBase *IntuitionBase;
struct Library *GadToolsBase;

static UWORD dri_Pens[] =
{0, 1, 1, 2, 1, 3, 1, 0, 3, ~0};

/* Gadtools menu stuff */
#define QUIT 1
struct NewMenu sdm_menu[] =
{
    {NM_TITLE, "Project", 0, 0, 0, 0,},
    {NM_ITEM, "Quit", "Q", 0, 0, (void *) QUIT,},
    {NM_END, NULL, 0, 0, 0, 0,},
};

struct EasyStruct failedES =
{
    sizeof(struct EasyStruct), 0, "AMPub",
    "%s",
    "OK",
};

void main(void);

void
main(void)
{

    struct Screen *pubscreen;
    struct Window *window;
    struct Menu *menu;
    void *vi;

    struct IntuiMessage *msg;
    UBYTE defaultpubname[MAXPUBSCREENNAME];
    UWORD oldstatus;
    UWORD oldmodes;
    ULONG signal, wsignal, lastsignal, allocsignal;
    ULONG oserror;
    BOOL ABORT = FALSE;
    BOOL CLOSEDOWN = FALSE;
    struct Task *task;

    /* Fails silently if not V36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36)) {
        if (GadToolsBase = OpenLibrary("gadtools.library", 36)) {
            if ((allocsignal = AllocSignal(-1)) != -1) {
                lastsignal = 1 << allocsignal;

                /* We'll use our task address to get signaled when the last
                 * visitor window leaves. */
                task =
                  (struct Task *)&((struct Process *)FindTask(NULL))->pr_Task;

                /* Get the name of the currently default
                 * public screen (normally WB) */
                GetDefaultPubScreen(defaultpubname);

                /* Note that no checking is done to make sure the
                 * public screen name is unique.  If the public name is
                 * not unique, OSERR_PUBNOTUNIQUE is returned in an
                 * application supplied variable as data for the SA_ErrorCode
                 * tag.  In this case however, if OpenScreen fails, for
                 * whatever reason, we just bail out.
                 */

                if (pubscreen = OpenScreenTags(NULL,
                                SA_DisplayID, HIRES_KEY,
                                SA_Depth, 2,
                                SA_Title,  "AmigaMail Test Public Screen",
                                SA_PubName,"AmigaMail Test Public Screen",

                                SA_PubSig, allocsignal,
                                SA_PubTask, task,
                                        /* Normally you'd supply 0L, if you
                                        * the task which opened the screen to
                                        * be signaled.
                                        */
                                SA_Pens, dri_Pens, SA_ErrorCode, &oserror,
                                TAG_END)) {
                    if (menu = CreateMenus(sdm_menu, TAG_DONE)) {
                        if (window = OpenWindowTags(NULL,
                                WA_Top, pubscreen->BarHeight + 1,
                                WA_Height, pubscreen->Height
                                        - (pubscreen->BarHeight + 1),
                                WA_CustomScreen, pubscreen,
                                WA_IDCMP, MENUPICK,
                                WA_Flags, ACTIVATE|BACKDROP|
                                        BORDERLESS|NOCAREREFRESH,
                                TAG_END)) {

                            vi = GetVisualInfo(pubscreen, TAG_END);
                            if (LayoutMenus(menu, vi, TAG_DONE)) {
                                SetMenuStrip(window, menu);

                                /* Now we have a screen, named "AmigaMail Test
                                 * Public Screen", which is still marked
                                 * private, but ready to be made public.
                                 */
                                oldstatus = PubScreenStatus(pubscreen, 0);
                                /* Now it is really public. We'll make it the
                                 * default (and shanghai attempts to open
                                 * windows on the Workbench.
                                 */
                                SetDefaultPubScreen(
                                        "AmigaMail Test Public Screen");

                                oldmodes = SetPubScreenModes(SHANGHAI|
                                                POPPUBSCREEN);

                                wsignal = 1 << window->UserPort->mp_SigBit;

                                do {
                                    signal = Wait(wsignal|lastsignal);
                                    /* The last visitor just left our screen */
                                    if (signal & lastsignal) {
                                        if (CLOSEDOWN)
                                            if ((oldstatus =
                                                PubScreenStatus(pubscreen,
                                                        PSNF_PRIVATE)) & 1)
                                                ABORT = TRUE;
                                    }

                                    /* If not wsignal, this while-loop will not
                                     * be executed.
                                     */
                                    while(msg = (struct IntuiMessage *)
                                          GetMsg(window->UserPort)) {
                                        if (msg->Class == MENUPICK) {
                                            if ((UWORD)(MENU_USERDATA
                                                    (ItemAddress(menu,
                                                        msg->Code))) == QUIT) {
                                                /* If we can't make our screen
                                                 * private again, we got
                                                 * visitor window(s). Note
                                                 * window(s) you open without
                                                 * specifiying SA_PubScreenName,
                                                 * are NOT considered visitor
                                                 * windows.  So it's easy to try
                                                 * to make your public screen
                                                 * private again, without having
                                                 * to close down your own windows
                                                 * first.
                                                 */

                                                if ((oldstatus =
                                                    PubScreenStatus(pubscreen,
                                                            PSNF_PRIVATE)) & 1) {
                                                    ABORT = TRUE;
                                                }
                                                else {
                                                    /* Close as soon as possible */
                                                    CLOSEDOWN = TRUE;
                                                    EasyRequest(window,
                                                            &failedES,
                                                            NULL,
"Can't make screen private.\nPlease close all windows");
                                                }
```

```
                                }
                            }
                            ReplyMsg((struct Message *)msg);
                        }
                    } while (ABORT == FALSE);

                    /* Restore orignal default */
                    SetDefaultPubScreen(defaultpubname);
                    ClearMenuStrip(window);
                    FreeVisualInfo(vi);
                } else
                    EasyRequest(NULL,
                                &failedES,
                                NULL,
                                "Can't layout menu's");
                CloseWindow(window);
            } else
                EasyRequest(NULL,
                            &failedES,
                            NULL,
                            "Can't open window");
            FreeMenus(menu);
        } else EasyRequest(NULL,
                           &failedES,
                           NULL,
                           "Can't create menu's");
        CloseScreen(pubscreen);
    } else
        EasyRequest(NULL,
                    &failedES,
                    NULL,
                    "Can't open public screen");
    FreeSignal(allocsignal);
} else
    EasyRequest(NULL, &failedES, NULL, "No signal available");
    CloseLibrary(GadToolsBase);
}
CloseLibrary(IntuitionBase);
}
```

```
/*
 * V36 Clone Workbench screen
 *
 * Compiled with SAS C 5.10:  lc -cfist -v -L CloneWB.c
 * Use V36 include files.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <graphics/displayinfo.h>

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/graphics_protos.h>

#include <pragmas/exec_pragmas.h>
#include <pragmas/intuition_pragmas.h>
#include <pragmas/graphics_pragmas.h>

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;

struct EasyStruct failedES = {
    sizeof(struct EasyStruct), 0, "CWB",
    "%s",
    "OK",
};

void            main(void);

void
main(void)
{
    struct Screen *wbscreen;
    struct Screen *clonescreen;
    struct Window *window;
    struct ViewPort *vp;
    struct IntuiMessage *msg;
    ULONG modeID;

    struct DisplayInfo displayinfo;
    struct MonitorInfo monitorinfo;
    struct DimensionInfo dimensioninfo;
    struct NameInfo nameinfo;
    struct DrawInfo *drawinfo;

    ULONG result;

    /* Fails silently when not V36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36)) {
        if (GfxBase = OpenLibrary("graphics.library", 36)) {

            if (wbscreen = LockPubScreen("Workbench")) {
                /* Using intuition.library/GetScreenDrawInfo(), we get the pen
                 * array we'll use for the screen clone the easy way. */
                drawinfo = GetScreenDrawInfo(wbscreen);

                vp = &(wbscreen->ViewPort);
                /* Use graphics.library/GetVPModeID() to get the ModeID of the
                 * Workbench screen. */
                if ((modeID = GetVPModeID(vp)) != INVALID_ID) {
                    /* OK, we got a valid ModeID for the Workbench
                     * screen (surprise) */

                    /* With just the ModeID, the lock on the Workbench
                     * screen & drawinfo->dri_Pens we're able to clone the
                     * Workbench exactly, displaymode and all. When you want
                     * to know everything there is to know about this
                     * displaymode, use graphics.library/GetDisplayInfoData().
                     */
                    /* 'result' is the actual # of bytes copies
                     * into the buffer.
                     * See graphics/displayinfo.h for definitions
                     * of the various structures.
                     */
                    result = GetDisplayInfoData(NULL,
                                                (UBYTE *)&monitorinfo,
                                                sizeof(struct MonitorInfo),
                                                DTAG_MNTR, modeID);
                    result = GetDisplayInfoData(NULL,
                                                (UBYTE *)&displayinfo,
                                                sizeof(struct DisplayInfo),
                                                DTAG_DISP, modeID);
                    result = GetDisplayInfoData(NULL,
                                                (UBYTE *)&dimensioninfo,
                                                sizeof(struct DimensionInfo),
```

```
                                        DTAG_DIMS,
                                        modeID);
                result = GetDisplayInfoData(NULL,
                                        (UBYTE *)&nameinfo,
                                        sizeof(struct NameInfo),
                                        DTAG_NAME,
                                        modeID);

                /* If you don't want to clone the workbench exactly,
                 * you could use one of the other overscan modes
                 * and/or min/max.
                 * dimensions in dimensioninfo or open a regular screen.
                 */
                if (clonescreen = OpenScreenTags(NULL,
                    /* Could use STDSCREENWIDTH/HEIGHT for non-scrollable
                     * screens. Those make */
                    SA_Width, wbscreen->Width,

                    /* OpenScreen() use the textoverscan values */
                    SA_Height, wbscreen->Height,

                    SA_DisplayID, modeID,

                    SA_Depth, wbscreen->BitMap.Depth,

                    /*Workbench always sizes up to OSCAN_TEXT, so do we */
                    SA_Overscan, OSCAN_TEXT,

                    SA_Pens, drawinfo->dri_Pens,

                    SA_Title, nameinfo.Name,

                    /* Doesn't hurt for screens which don't scroll */
                    SA_AutoScroll, TRUE,

                    TAG_END)) {

                    if (window = OpenWindowTags(NULL,
                        WA_Top, clonescreen->BarHeight + 1,
                        WA_Height, clonescreen->Height
                                        - (clonescreen->BarHeight + 1),
                        WA_CustomScreen, clonescreen,
                        WA_MinWidth, 320,
                        WA_MinHeight, 100,
                        WA_MaxWidth, clonescreen->Width,
                        WA_MaxHeight, clonescreen->Height,

                        /* I'm only interested in CLOSEWINDOW messages */
                        WA_IDCMP, CLOSEWINDOW,
                        WA_Flags, WINDOWSIZING|WINDOWDRAG|
                                WINDOWDEPTH|WINDOWCLOSE|ACTIVATE,
                        WA_Title, "Close to exit.",
                        TAG_END)) {

                        /* Wait for the closewindow message */
                        WaitPort(window->UserPort);
                        /* And remove message from the port */
                        while(msg = (struct IntuiMessage *)
                                GetMsg(window->UserPort))
                            ReplyMsg((struct Message *)msg);
                        CloseWindow(window);
                    } else
                        EasyRequest(NULL, &failedES, NULL,
                                "Can't open window");
                    CloseScreen(clonescreen);
                } else
                    EasyRequest(NULL, &failedES, NULL,
                                "Can't open screen");
            } else
                EasyRequest(NULL, &failedES, NULL, "Invalide ModeID");
            FreeScreenDrawInfo(wbscreen, drawinfo);
            UnlockPubScreen(NULL, wbscreen);
        } else
            EasyRequest(NULL, &failedES, NULL,
                "Can't lock Workbench screen");
        CloseLibrary(GfxBase);
    }
    CloseLibrary(IntuitionBase);
}
```

```
/*
 * V36 Screen Displaymode selector example
 *
 * Compiled with SAS C 5.05:  lc -cfist -v -L ScreenDisplayModes.c
 * Use V36 include files.
 */

#include <string.h>
#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <libraries/gadtools.h>
#include <graphics/displayinfo.h>
#include <graphics/text.h>
#include <exec/memory.h>
#include <exec/ports.h>

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/graphics_protos.h>
#include <clib/gadtools_protos.h>
#include <clib/alib_protos.h>

#include <pragmas/exec_pragmas.h>
#include <pragmas/intuition_pragmas.h>
#include <pragmas/graphics_pragmas.h>
#include <pragmas/gadtools_pragmas.h>

#define SWITCH 888
#define QUIT   999

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Library *GadToolsBase;

void main(void);
void ShowDisplayModes(struct List * dlist);
struct Screen *OpenAScreen(struct DimensionInfo * di, UBYTE * name,
                           ULONG overscantype);
struct Window *OpenAWindow(struct Screen * screen, ULONG overscantype);

struct DisplayNode {
    struct Node dn_Node;
    struct DimensionInfo dn_Dimensioninfo;
};

/* The workbench compatible pens we'll use for that New Look. */
static UWORD dri_Pens[] = {0, 1, 1, 2, 1, 3, 1, 0, 3, ~0};

/* Gadtools menu stuff */
struct NewMenu sdm_menu[] =
{
    {NM_TITLE, "Project", 0, 0, 0, 0,},
    {NM_ITEM, "Switch Mode", "S", 0, 0, (void *) SWITCH,},
    {NM_ITEM, "Set Overscan", 0, 0, 0, 0,},
    {NM_SUB, "Text", 0, CHECKIT | CHECKED, ~1, (void *) OSCAN_TEXT,},
    {NM_SUB, "Standard", 0, CHECKIT, ~2, (void *) OSCAN_STANDARD,},
    {NM_SUB, "Max", 0, CHECKIT, ~4, (void *) OSCAN_MAX,},
    {NM_SUB, "Video", 0, CHECKIT, ~8, (void *) OSCAN_VIDEO,},
    {NM_ITEM, NM_BARLABEL, 0, CHECKIT, 0, 0,},
    {NM_ITEM, "Quit", "Q", 0, 0, (void *) QUIT,},
    {NM_END, NULL, 0, 0, 0, 0,},
};

UBYTE *OScanDescr[] = {NULL, "OSCAN_TEXT", "OSCAN_STANDARD",
                       "OSCAN_MAX", "OSCAN_VIDEO"};

struct EasyStruct failedES =
{
    sizeof(struct EasyStruct), 0, "SDM",
    "%s",
    "OK",
};

void
main(void)
{
    struct List *dlist;
    struct DisplayNode *dnode;
    struct DisplayNode *wnode, *nnode;

    ULONG modeID;
    ULONG skipID;

    ULONG result;
    struct DisplayInfo displayinfo;
    struct NameInfo nameinfo;

    /* Fails silently if not V36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36)) {
        if (GfxBase = OpenLibrary("graphics.library", 36)) {
```

```
    if (GadToolsBase = OpenLibrary("gadtools.library", 36)) {
        if (dlist = AllocMem(sizeof(struct List), MEMF_CLEAR)) {
            NewList(dlist);

            /*
             * Don't want duplicate entries in the list for the
             * 'default monitor', so we'll skip the the videomode
             * for which default.monitor is the alias.
             */

            /* INVALID_ID indicates the beginning and the end
            * of the list of available keys.
            */
            modeID = INVALID_ID;

            GetDisplayInfoData(NULL, (UBYTE *) & displayinfo,
                               sizeof(struct DisplayInfo),
                               DTAG_DISP, LORES_KEY);
            if (displayinfo.PropertyFlags & DIPF_IS_PAL)
                skipID = PAL_MONITOR_ID;
            else
                skipID = NTSC_MONITOR_ID;
            while ((modeID = NextDisplayInfo(modeID)) != INVALID_ID) {
                if ((modeID & MONITOR_ID_MASK) != skipID) {
                  /*
                   * For this example,only 'named' keys are accepted.  Others
                   * which have no description are left out
                   * even though they may be available.
                   * HAM and EXTRAHALFBRIGHT are examples of this.
                   * If needed a name could be made like '320x400 HAM Interlace'.
                   */

                    if (result = GetDisplayInfoData(NULL,
                         (UBYTE *) & nameinfo,
                         sizeof(struct NameInfo),
                         DTAG_NAME,
                         modeID)) {
                        result = GetDisplayInfoData(NULL,
                         (UBYTE *) & displayinfo,
                         sizeof(struct DisplayInfo),
                         DTAG_DISP,
                         modeID);
                        if (!(displayinfo.NotAvailable)) {
                            if (dnode =
                     (struct DisplayNode *)
                     AllocMem(sizeof(struct DisplayNode),
                      MEMF_CLEAR)) {
                                result = GetDisplayInfoData(NULL,
                       (UBYTE *) &
                       (dnode->dn_Dimensioninfo),
                       sizeof(struct DimensionInfo),
                       DTAG_DIMS,
                       modeID);
                                /* to keep it short: if NOMEM,
                     * just don't copy
                     */
                                if (dnode->dn_Node.ln_Name
                        = AllocMem(strlen(nameinfo.Name)
                            + 1, MEMF_CLEAR))
                                    strcpy(dnode->dn_Node.ln_Name,
                       nameinfo.Name);
                                AddTail(dlist, (struct Node *) dnode);
                            } else {
                                EasyRequest(NULL, &failedES, NULL,
                             "Out of memory");
                                /* Force modeID to INVALID to break */
                                modeID = INVALID_ID;
                            }
                        }
                    }
                }
            }
            ShowDisplayModes(dlist);

            wnode = (struct DisplayNode *) dlist->lh_Head;
            while (nnode = (struct DisplayNode *)
                   (wnode->dn_Node.ln_Succ)) {

                if (wnode->dn_Node.ln_Name)
                    FreeMem(wnode->dn_Node.ln_Name,
                            strlen(wnode->dn_Node.ln_Name) +1);
                Remove((struct Node *) wnode);
                FreeMem(wnode, sizeof(struct DisplayNode));
                wnode = nnode;
            }
            FreeMem(dlist, sizeof(struct List));
        } else
            EasyRequest(NULL, &failedES, NULL, "Out of memory");
        CloseLibrary(GadToolsBase);
    }
```

```
        CloseLibrary(GfxBase);
    }
    CloseLibrary(IntuitionBase);
  }
}

void
ShowDisplayModes(struct List * dlist)
{
    struct Screen *screen;
    struct Window *window;
    struct Gadget *glist, *gadget, *hitgadget;
    struct DrawInfo *drawinfo;
    struct TextFont *defaultfont;
    struct TextAttr *textattr;
    struct IntuiMessage *imsg;
    struct NewGadget *ng;
    struct Menu *menu;
    struct MenuItem *item;
    void *vi;
    ULONG iclass, icode, code;
    struct DisplayNode *dnode;
    struct DimensionInfo *dimensioninfo;
    ULONG overscantype = OSCAN_TEXT;
    ULONG curmode = 0;
    BOOL ABORT = TRUE, OK;
    int i;


    if (ng = AllocMem(sizeof(struct NewGadget), MEMF_CLEAR)) {
        if (textattr = AllocMem(sizeof(struct TextAttr), MEMF_CLEAR)) {
            if (textattr->ta_Name = AllocMem(48, MEMF_CLEAR)) {

                dnode = (struct DisplayNode *) dlist->lh_Head;

                if (menu = CreateMenus(sdm_menu, TAG_DONE)) {
                    do {
                        dimensioninfo = &(dnode->dn_Dimensioninfo);
                        OK = FALSE;
                        if (screen = OpenAScreen(dimensioninfo,
                              dnode->dn_Node.ln_Name,
                              overscantype)) {
                            drawinfo = GetScreenDrawInfo(screen);
                            defaultfont = drawinfo->dri_Font;
                            strcpy(textattr->ta_Name,
                                defaultfont->tf_Message.mn_Node.ln_Name);

                            textattr->ta_YSize = defaultfont->tf_YSize;
                            textattr->ta_Style = defaultfont->tf_Style;
                            textattr->ta_Flags = defaultfont->tf_Flags;

                            if (window = OpenAWindow(screen, overscantype)) {
                                vi = GetVisualInfo(screen, TAG_END);
                                if (LayoutMenus(menu, vi, TAG_DONE)) {
                                    if (gadget = CreateContext(&glist)) {
                                        ng->ng_LeftEdge =
                                    window->BorderLeft + 10;
                                        ng->ng_TopEdge =
                                    window->BorderTop + 10;
                                        ng->ng_Width =
                                    window->Width -
                                      (window->BorderLeft + 10) -
                                        (window->BorderRight + 10);
                                        ng->ng_Height =
                                    window->Height -
                                      (window->BorderTop + 10) -
                                        (window->BorderBottom + 10) -
                                  (4 + defaultfont->tf_YSize + 2);
                                        ng->ng_TextAttr = textattr;
                                        ng->ng_GadgetText = NULL;
                                        ng->ng_VisualInfo = vi;
                                        ng->ng_GadgetID = 1;
                                        ng->ng_Flags = PLACETEXT_ABOVE;
                                        gadget =
                                    CreateGadget(LISTVIEW_KIND, gadget,
                                         ng, GTLV_Labels, dlist,
                                         GTLV_ShowSelected, NULL,
                                         GTLV_Selected, curmode,
                                         TAG_END);

                                        ng->ng_TopEdge +=
                                    gadget->Height +
                                      defaultfont->tf_YSize + 10;
                                        ng->ng_Width = 80;
                                        ng->ng_LeftEdge =
                                    ((window->Width - window->BorderLeft
                                      - window->BorderRight) / 2) - 30;
                                        ng->ng_Height = defaultfont->tf_YSize + 8;
                                        ng->ng_GadgetID = 2;
                                        ng->ng_GadgetText = "OK";
```

```
                                ng->ng_VisualInfo = vi;
                                ng->ng_Flags = PLACETEXT_IN;
                                gadget =
                            CreateGadget(BUTTON_KIND, gadget,
                                ng, TAG_END);

                                AddGList(window, glist, -1, -1, NULL);
                                RefreshGList(glist, window, NULL, -1);
                                GT_RefreshWindow(window, NULL);
                                SetMenuStrip(window, menu);
                                ABORT = FALSE;
                            } else
                                EasyRequest(window, &failedES, NULL,
                             "Can't create gadget context");
                            } else
                                EasyRequest(window, &failedES, NULL,
                           "Can't layout menus");

                            do {
                                WaitPort(window->UserPort);
                                while (imsg = GT_GetIMsg(window->UserPort)) {
                                    iclass = imsg->Class;
                                    icode = imsg->Code;
                                    hitgadget =
                            (struct Gadget *) imsg->IAddress;
                                    GT_ReplyIMsg(imsg);

                                    switch (iclass) {
                                    case GADGETUP:
                                        if (hitgadget->GadgetID == 1) {
                                            dnode =
                            (struct DisplayNode *)
                             dlist->lh_Head;
                                            for (i = 0; i < icode; i++)
                                            dnode =
                            (struct DisplayNode *)
                             dnode->dn_Node.ln_Succ;
                                            curmode = i;
                                        }
                                        if (hitgadget->GadgetID == 2)
                                            OK = TRUE;
                                        break;
                                    case MENUPICK:
                                        while ((icode != MENUNULL)
                             && (ABORT == FALSE)) {
                                            item = ItemAddress(menu, icode);

                                            code = (ULONG) MENU_USERDATA(item);
                                            if (code == QUIT)
                                                ABORT = TRUE;
                                            else if
                                                (code == SWITCH)
                                                OK = TRUE;
                                            else
                                                overscantype = (ULONG) code;
                                            icode = item->NextSelect;
                                        }
                                        break;
                                    case CLOSEWINDOW:
                                        ABORT = TRUE;
                                        break;
                                    }
                                }
                            } while (ABORT == FALSE && OK == FALSE);
                            ClearMenuStrip(window);
                            CloseWindow(window);
                            FreeVisualInfo(vi);
                            FreeGadgets(glist);
                        } else
                            EasyRequest(NULL, &failedES, NULL,
                             "Can't open window");
                        FreeScreenDrawInfo(screen, drawinfo);
                        CloseScreen(screen);
                    } else
                        EasyRequest(NULL, &failedES, NULL,
                         "Can't open screen");
                } while (ABORT == FALSE);
                FreeMenus(menu);
            } else
                EasyRequest(NULL, &failedES, NULL, "Can't create menus");
            FreeMem(textattr->ta_Name, 48);
        } else
            EasyRequest(NULL, &failedES, NULL, "Out of memory");
        FreeMem(textattr, sizeof(struct TextAttr));
    } else
        EasyRequest(NULL, &failedES, NULL, "Out of memory");
    FreeMem(ng, sizeof(struct NewGadget));
} else
    EasyRequest(NULL, &failedES, NULL, "Out of memory");
}
```

```
/*
 * It's advised to use one of the overscan constants and
 * STDSCREENWIDTH and STDSCREENHEIGHT. For this example however, we'll
 * skip all that an use QueryOverscan to get the rectangle describing the
 * requested overscantype and pass that as the displayclip description.
 * Actually, since we pass the standard rectangle from the display database,
 * it is equivalent to the prefered:
 *
 * screen = OpenScreenTags(NULL,
 *                         SA_DisplayID, di->Header.DisplayID,
 *                         SA_Overscan, overscantype,
 *                         SA_Width, STDSCREENWIDTH,
 *                         SA_Height, STDSCREENHEIGHT,
 *                         SA_Title, name,
 *                         SA_Depth, 2,
 *                         SA_SysFont, 1,
 *                         SA_Pens, dri_Pens,
 *                         TAG_END);
 */

struct Screen *
OpenAScreen(struct DimensionInfo * di, UBYTE * name, ULONG overscantype)
{

    struct Rectangle rectangle;

    /* Can't fail, already made sure it's a valid displayID */
    QueryOverscan(di->Header.DisplayID, &rectangle, overscantype);


    return (OpenScreenTags(NULL,
                           SA_DisplayID, di->Header.DisplayID,
                           SA_DClip, &rectangle,
                           SA_Title, name,
                           SA_Depth, 2,
                           SA_SysFont, 1,   /* Use the prefered WB screen font */
                           SA_Pens, dri_Pens,
                           TAG_END));
}

struct Window *
OpenAWindow(struct Screen * screen, ULONG overscantype)
{
    return (OpenWindowTags(NULL,
                           WA_Top, screen->BarHeight + 1,
                           WA_Height, screen->Height - (screen->BarHeight + 1),
                           WA_CustomScreen, screen,
                           WA_IDCMP, CLOSEWINDOW | LISTVIEWIDCMP | MENUPICK,
                           WA_Flags, WINDOWCLOSE | ACTIVATE,
                           WA_Title, OScanDescr[overscantype],
                           TAG_END));
}
```

❖

Amiga Mail
Volume II
September/October 1990

Master Index for Amiga Documentation (Preliminary)

This index is designed to simplify extracting information from Amiga technical documentation. It indexes Amiga technical documentation available from CATS as well as other sources, including the RKMs and Bantam's *AmigaDOS Manual.* As this index is preliminary, it is subject to change without notice. It is provided here to elicit feedback from developers and to furnish subscribers with a master index with the least possible delay. Comments and suggestions should be sent to:

Ken Farinsky - CATS
Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380

Responses can also be sent via e-mail to:

ken@cbmvax.commodore.com or ...{uunet| rutgers}!cbmvax!ken

All index entries are indexed by either by *KEY.chapter number.subheading number* or *KEY.disk number.directory number*, where KEY is one of he following:

86DN	1986 Devcon Notes (No longer available)
88DN	1988 Devcon Notes
88d	1988 Devcon Disks (4 disks)
89DN	1989 Devcon Notes
89d	1989 Devcon Disks (2 disks)
90DN	1990 Devcon Notes
90d	1990 Devcon Disks (3 disks)
A1000	A1000 Schematics and Expansion Specs
ADOS	AmigaDOS Manual 2nd Edition (Bantam Books ISBN 0-553-34440-0)
AMT1	AmigaMail Tech Notes (Volume 1)
EN13	1.3 Enhancer Manual
HW	Amiga Hardware Manual (Addison-Wesley ISBN 0-201-18157-6)
IA	Amiga RKM Includes and Autodocs (Addison-Wesley ISBN 0-201-18177-0)
IFFd	EA IFF Disk 1988 (also Fish Disk 185)
IFF	IFF Manual 1988 (also in 1.3 Includes and Autodocs book)
LDd	Addison-Wesley Companion Disks (lharcs on Fish Disk 344)
LD	Amiga RKM Libraries and Devices (Addison-Wesley ISBN 0-201-18187-8)
ND13	1.3 Native Developer Update (4 disks)
ND20	2.0 Native Developer Update (4 disks)
SWT	Software Toolkit (1988)
TREF	A500/A2000 Technical Reference Manual

1.3.3
  Enforcer, update (1.3) 90d.3.14
1.3
  assembly prefix IA.H
  autoboot 88DN.13
  autodocs IA.A, ND13.1
  c lang. cross reference IA.H
  changes from 1.2 88DN.13
  changes to printer.device AMT1.10.4
  devices IA.B
  function index IA.J
  hard disk installation AMT1.2.7, AMT1.2.8
  include files IA.D
  includes ND13.2.1
  libraries IA.A
  linker libraries IA.F
  LVO reference IA.H
  preferences 88DN.29, AMT1.10.2
  printer driver writing 88DN.21
  printer.device changes 88DN.18
  printer.device 88DN.17
  resources IA.C
  sample device IA.G
  sample library IA.G
  structure offsets IA.H
1.4
  See also 2.0
  commodities 89DN.18
  compatibility 89DN.22
  display modes 89DN.6
  DOS features 89DN.17, 89DN.23
  ECS support 89DN.4
  exec 89DN.20
  features (preliminary) 89DN.2
  intuition features 89DN.12
  preferences 89DN.15
  workbench features 89DN.3
2.0
  autodocs disk ND20.1
  commodities 90DN.25
  compatibility 90DN.16
  dos.library functions 90DN.31
  examples disk ND20.3
  exec 90DN.34
  features 90DN.23
  font-size accomodation 90DN.37
  gadtools 90DN.30
  includes disk ND20.2.1
  intuition boopsi 90DN.32
  intuition 90DN.29
  overview 90DN.19
  tools disk ND20.3.9
  workbench 90DN.24
32 bit
  software compatibility 88DN.27
34010
  See also A2410
  A2410 hires card 90DN.35
68000
  cards reference tool SWT.9
  identifying AMT1.3.5
68010
  compatibility AMT1.1.2, LD.Preface
  identifying AMT1.3.5
68020
  compatibility 88DN.27, AMT1.1.11, AMT1.1.2, LD.appxA, LD.Preface
  identifying AMT1.3.5
68030
  compatibility 88DN.27, LD.appxA, LD.Preface
  hardware design AMT1.12.5
  identifying AMT1.3.5
68881
  adding resource AMT1.13.1
7 wire
  serial LD.40.7
8520
  See also CIA
  spec (hardware) HW.appxF
8SVX
  See also IFF
  CHAN chunk spec IFF.5.7
  CHAN spec IFFd.7
  IFF spec IFF.3.4
  in IFF AMT1.7.2
  PAN chunk spec IFF.5.7
  PAN spec IFFd.7
  player AMT1.7.3
A1000
  audio cutoff filter AMT1.6.4
  autoconfig A1000.2.3
  backplane design A1000.2.2, A1000.2.4
  board design A1000.2.3
  board ID information A1000.2.3
  connectors (hardware) TREF.1
  connectors A1000.1.2
  diagrams A1000.appxA
  expansion architecture A1000.2.1
  expansion power budget A1000.2.4
  expansion timing A1000.2.4
  PAL equations A1000.1.1
  ports (hardware) TREF.1
  ports A1000.1.2
  schematics A1000.appxA
A2000
  audio cutoff filter AMT1.6.4
  block diagram (hardware) TREF.2
  connectors (hardware) TREF.1
  diagrams TREF.appxA
  expansion board design TREF.3
  expansion signals TREF.3.4
  mechanicals TREF.appxA.2
  motherboards AMT1.14.4
  PAL equations TREF.3.1
  PAL/NTSC switch AMT1.1.10
  peripheral design 89DN.8
  ports (hardware) TREF.1
  power budget TREF.7.2
  schematics TREF.appxA.2, TREF.appxB
  video slot TREF.3.4
A2024
  readmes ND13.4.5
  support (1.4) 89DN.6.1
A2410
  hires graphics card 90DN.35
  introduction 89DN.10
A2620
  compatibility 89DN.13
A2630
  compatibility 89DN.13
A3000
  adding RAM AMT1.12.7
  BIGRAM 90DN.3
  compatibility (2.0) 90DN.16
  expansion slot 90DN.1
  local bus expansion 90DN.1
  new products 1990 90DN.19
  system architecture 90DN.36
  Zorro III design 90DN.3
  Zorro III Spec 90DN.2
A500
  block diagram (hardware) TREF.2
  connectors (hardware) TREF.1
  diagrams TREF.appxA
  expansion board design 88DN.33, TREF.3
  expansion signals TREF.3.4
  hardware interface 88DN.33
  mechanicals TREF.appxA.2
  peripheral design 89DN.8
  ports (hardware) TREF.1
  power budget TREF.7.2
  schematics TREF.appxA.2, TREF.appxB
AbortIO
  function description LD.19.3
ACBM
  See also IFF
  IFF spec IFF.5.1, IFFd.7
Access mode
  boopsi 90DN.32
Activation flags
  gadget LD.4.4
Address
  arexx 89DN.11
AddTOF
  amiga.lib LD.appxB.2
afp
  amiga.lib LD.appxB.2
Agnus
  See also Chip
  pin assignment (hardware) HW.appxC
Alert
  68000 guru numbers LD.17.8
  Amiga guru numbers (exec/alerts.h) IA.D
  dead-end LD.6.2
  examples (alert) LDd.1.8
  examples (displayalert) LDd.1.8
  intuition LD.6.2
  recovery LD.6.2
Aliasing
  audio (hardware) HW.5.4
Alignment
  structure 89DN.20, 90DN.34, LD.Preface
ALink
  See also Linker
Allocate
  audio LD.31.2
  layer LD.26.2
  memory LD.15
  sprite LD.25.2
Alphanumeric
  printer LD.39.4
Amiga key
  style LD.11.4
Amiga.lib
  autodocs (1.3) IA.F
  electronic (1.3) ND13.3.7
  exec support listings 88d.4.3
  general LD.appxB.2
AmigaDOS
  See also Dos.library, DOS
  developer docs ADOS.2
  technical docs ADOS.3
  user docs ADOS.1
AmigaVision
  authoring hints 90DN.22
  construction set 90DN.6
  general 90DN.6
  standards 90DN.17
Amix
  See also Unix
  A2410 hires card 90DN.35
  Amiga graphics 89DN.16
  device drivers 89DN.19
  examples (device driver) 89DN.19
  examples (par.c) 89d.2.14
  portability 89DN.5
Analog
  from digital (audio) AMT1.7.3
ANBM
  See also IFF
  IFF spec IFF.5.2, IFFd.7
ANHD
  See also IFF
  ANIMbrush format 90DN.11
  IFF 88DN.3
Animation
  See also ANIM, Bob, Double buffered, Gel
  ANIMbrush 90DN.11
  collision AMT1.5.6, LD.25.8
  examples (anim.c (1.3)) LDd.1.3
  examples (animtools.c (1.3)) LDd.1.3
  examples (anim_ex.c (1.3)) LDd.1.3
  examples (bob.c (1.3)) LDd.1.3
  examples (collide.c) AMT1.5.6
  examples (complete) LD.25.11
  examples (ssprite.c (1.3)) LDd.1.3
  examples (vsprite.c (1.3)) LDd.1.3
  gel initialization LD.25.1
  IFF spec 88DN.3
  methods LD.25.1, LD.25.10
ANIM
  See also IFF
  ANIMbrush format 90DN.11
  examples (ShowAnim) 88d.2.2
  GrabAnim 88d.2.2
  IFF spec 88DN.3, IFF.5.3, IFFd.7
  ShowAnim 88d.2.2
ANSI

CON: and RAW: codes ADOS.2.5
escape sequence LD.33.6, LD.33.9
AppBuilder
overview 90DN.17
AppIcon
description (2.0) 90DN.24
examples (appicon.c) ND20.3.1
examples (appicon.image) ND20.3.1
in AppShell 90DN.28
AppMenu
description (2.0) 90DN.24
examples (appmenu.c) ND20.3.1
in AppShell 90DN.28
AppShell
autodocs (preliminary 2.0) 90d.2.5
docs (preliminary 2.0) 90DN.28
examples (projlist (2.0)) 90d.2.5
examples (watchman.c (2.0)) 90d.2.5
includes (preliminary 2.0) 90d.2.5
lib (preliminary 2.0) 90d.2.5
overivew 90DN.17
AppWindow
description (2.0) 90DN.24
examples (appwindow.c) ND20.3.1
in AppShell 90DN.28
Arbitration
audio LD.31.2
between processes AMT1.2.12
Archive
preferences (2.0) 90DN.10
Arcnet
card 90DN.4
Area fill
See also Fill
ARexx
design considerations 90DN.27
docs (1989) 89d.2.12
event processing AMT1.4.5
examples (minrexx.zoo (1989)) 89d.2.12
examples (simple tools) 90DN.27
examples (simplerexx.c) AMT1.14.5
implementing 90DN.27
in AppShell 90DN.28
interfacing to 89DN.11
midi command set (preliminary) AMT1.14.6
minrexx.zoo (1989) 89d.2.12
simple interface AMT1.14.5
Argument
arexx 89DN.11
command line processing (2.0) 90DN.31
DOS execute ADOS.1.2
examples (pattern.c) ND20.3.4
examples (readargs.c) ND20.3.4
workbench arg LD.30.4
arnd
amiga.lib LD.appxB.2
Asl.library
examples (asltest.c) ND20.3.5
overview (1.4) 89DN.23
Aspect
printer AMT1.10.3, LD.39.3
Assembler
68000 cards tool SWT.9
C language interface 88DN.20
calling libraries 88DN.20
compatibility 88DN.20
debugging 88DN.20
development rules LD.Preface
device IO AMT1.3.2
examples (blitter) 88d.1.4
examples (clicolors (packet)) AMT1.2.10
examples (debug) 88d.1.4
examples (device IO) AMT1.3.2
examples (general) 88d.1.4
examples (intserver) 88d.1.4
examples (open library) IA.A
examples (Rstartup.asm) 88d.3.3
examples (spp) 88d.3.2
examples (startup code) LD.30.6
examples (subroutines) 88d.1.4
examples (vertical blank interrupt) 88DN.20
guidelines IA.Preface
include files (1.3) IA.E
include files (using) 88DN.20
list macros LD.16.4
Metacomco ADOS.2.3
program environment ADOS.2.1
programming 88DN.20
structure definitions (1.3) IA.E
structure prefix list IA.H
structures 88DN.20
tips 88DN.20
Asterisk (*)
CLI current window ADOS.2.5
Asynchronous
device requests LD.19.5
Attribute
boopsi 90DN.32
Audio.device
See also Audio
Audio
8SVX player AMT1.7.3
aliasing (hardware) HW.5.4
articles AMT1.6
audio.device 88DN.12, AMT1.6.2, LD.31
clock constant AMT1.6.1
compatibility AMT1.1.10
cutoff filter AMT1.6.4
data rate (hardware) HW.5.2
debugging with AMT1.6.5
examples (atools.c) 88d.2.1
examples (audio cutoff filter) AMT1.6.4
examples (audio) 88DN.12
examples (audio.device I/O) AMT1.6.2
examples (audio1.c (1.3)) LDd.2.11
examples (audio2.c (1.3)) LDd.2.11
examples (audiotools3.c) 88d.2.1
examples (convertsound.c) AMT1.6.6
examples (debtones.asm) AMT1.6.5
examples (double-buffered) LD.31.2
examples (simple audio) LD.31.2
examples (touch tone sound) AMT1.6.3
hardware HW.5
interrupts (hardware) HW.7.5
low level programming HW.5
low-pass filter (hardware) HW.5.4
Macintosh sounds AMT1.6.6
modulation (hardware) HW.5 3
PAL and NTSC differences AMT1.1.4, AMT1.6.1
sample size AMT1.7.3
scale (hardware) HW.5.6
SMUS player AMT1.7.2
split samples AMT1.7.3
tips and techniques 88DN.12
touch tones AMT1.6.3
volume (hardware) HW.5.7
Auto-Scroll
intuition (2.0) 90DN.29
Autoboot
See also Autoconfig, Expansion
articles AMT1.12
board design 88DN.25
byte-wide roms AMT1.12.1
description AMT1.12.3
drivers LD.27.3
examples (DemoBootRomCode) 88d.3.4
examples (loadseg.asm) 88d.3.4
expansion.library LD.27
mechanism 88DN.13
readmes (1.3) ND13.4.6
rigid disk block 90DN.33
Autoconfig
See also Autoboot, Expansion
and autoboot 88DN.25
articles AMT1.12
board design TREF.3.1
board ID information (hardware) HW.appxG.2
board ID information TREF.3.1
board lister 90d.3.7
debugging (hardware) HW.appxG.1
drivers TREF.3.2
expansion.library LD.27
expansion 88DN.24
PAL equations TREF.3.1
peripheral design 89DN.8
specification (hardware) HW.appxG
Zorro III ram board 90DN.3
Zorro III 90DN.2
Autodocs
device (1.3) IA.B
DOS (1.4) 89DN.17.1
electronic (1.3) ND13.1
electronic (2.0) ND20.1
generating 90d.3.9, ND20.3.18
library (1.3) IA.A
resource (1.3) IA.C
AutoRequest
buildsysrequest LD.6.1
requester LD.6.1
Available
memory LD.15.3
Backdrop
layer LD.26.3
window type LD.3.1
Backfill
intuition hook (2.0) 90DN.29
Backplane
A1000 A1000.2.2
expansion 88DN.24
Baseline
text LD.24.2
Beam position
detection (hardware) HW.7.4
Beep
display (visual) LD.10.7
BeginIO
amiga.lib LD.appxB.2
function description LD.19.3
Benchmarks
Math IEEE 88d.3.2
math IEEE 88DN.28
Bfputs
amiga.lib LD.appxB.2
Bitmap
block (FFS) AMT1.2.4
block (trackdisk) AMT1.8.1
display LD.2.3, LD.23.1
extension (FFS) AMT1.2.4
Usage LD.23.2
BLink
See also Linker
Blitter
blitlab tool 88d.4.1
copper control (hardware) HW.2.8
copy (hardware) HW.6.7
copy 88DN.11
DMA (hardware) HW.6.3
DMA 88DN.11
done flag (hardware) HW.6.9
ECS extensions (1.4) 89DN.4.1
examples (assembler) 88d.1.4
examples (memory clear) HW.6.17
examples (rotate bits) HW.6.17
examples (simple line) HW.6.17
examples (various) 88d.2.4
fill (hardware) HW.6.8
fill 88DN.11
function (hardware) HW.6.4
hardware HW.6
interrupt (hardware) HW.6.10
interrupts (hardware) HW.7.5
line drawing 88DN.11
line mode (hardware) HW.6.13
low level programming HW.6
mask (hardware) HW.6.5
mask 88DN.11
minterms (hardware) HW.6.4
modulo (hardware) HW.6.3
multitasking (hardware) HW.6.9
multitasking LD.23.4
programming 88DN.11, AMT1.5.3
shift (hardware) HW.6.5
shift 88DN.11
simulation 88DN.11
stencil LD.23.4
takeover 88DN.9
template LD.23.4
venn diagram (hardware) HW.6.4

Block
  size (query) AMT1.2.6
BMHD
  See also IFF
  introduction AMT1.7.1
Bob
  animation LD.25.7
  examples (use bob) LD.25.7
BODY
  See also IFF
  introduction AMT1.7.1
Bold
  text LD.24.6
Boolean
  gadget LD.4.3
BoolInfo
  gadget LD.4.4
Boopsi
  classes (2.0) 90DN.29
  examples (2.0) 90d.1.5, ND20.3.3
  object oriented intuition 90DN.32
Boot
  block (FFS) AMT1.2.4
  CDTV 90DN.8
  expansion LD.27.3
  floppy disk LD.appxC
  keyboard error code (hardware) HW.appxH.5
  preferences (2.0) 90DN.10
  rigid disk block 90DN.33
  virus AMT1.14.1
Borderless
  window type LD.3.1
Border
  border structure LD.8.1
  collision LD.25.8
  color LD.8.1
  coordinates LD.8.1
  description LD.8.1
BPTR
  definition ADOS.3.3
Bridgeboard
  See also Janus
  amiga memory access 89DN.7.1
  autodocs (1989) 89DN.7.2
  BIOS TREF.4.2
  dual port memory 89DN.7.1
  hardware TREF.4.1
  janus.library 89DN.7
  software (1986) TREF.4.3
Brush
  AmigaVision 90DN.22
  ANIMbrush format 90DN.11
  bob LD.25.7
BSTR
  definition ADOS.3.3
Buffer
  trackdisk LD.42.2
Bug reports
  submission HW.1, LD.Preface
  Zorro III 90DN.2
Buster
  See also Chip
  tower (A2000 expansion) 88DN.24
Busy loop
  better practices AMT1.9.4
Bus
  68000 (hardware) AMT1.12.8
Button
  gadtools (2.0) 90DN.30
C language
  assembler interface 88DN.20
  compiling examples LD.Preface
  development rules LD.Preface
  examples (open library) IA.A
  examples (reentrant code) 88DN.2
  examples (simple C) 88d.2.3
  guidelines IA.Preface
  include files (1.3) IA.D
  introduction 88DN.37
  library creation AMT1.3.4
  program environment ADOS.2.1
  programming tips 88DN.2
  quick start 89DN.1
  reentrant code 88DN.2
  structure cross reference IA.H
  structure definitions (1.3) IA.D
  support functions LD.appxB.2
Cache
  CPU compatibility 88DN.27
  library pointers LD.14.2
Callback
  hooks (2.0) 90DN.29
CAMG
  See also IFF
  introduction AMT1.7.1
  storage format (2.0) 90DN.11, AMT1.7.5
Capsable
  keymap table LD.33.11
CATS
  developer support LD.Preface
  documentation ordering AMT1.1.3
  information HW.1, IA.Preface
  manufacturer number AMT1.12.4
  ordering AMT1.7.2
CAT
  iff definition IA.I
CD-ROM
  See also CDTV
  characteristics 90DN.9
  emulation 90DN.12
  product concepts 90DN.9
  product design 90DN.9
  software 90DN.9
CDTV
  See also CD-ROM
  CD-ROM software 90DN.9
  CTrac 90DN.12
  development 90DN.12
  specification 90DN.7
  user interface 90DN.8
Changes
  disk (diskchange) LD.42.10
CHANGEWINDOW
  IDCMP (2.0) 90DN.29
Channel
  audio hardware HW.5.2
  audio LD.31.1
Character set
  amiga LD.33.3
Character
  international compatibility AMT1.1.4
  table of ADOS.2.5
Checkbox
  gadtools (2.0) 90DN.30
CheckIO
  function description LD.19.3
Checkmark
  menu LD.5.1
Checksum
  block (trackdisk) AMT1.8.1
Chip
  8520 TREF.6.2
  Agnus TREF.6.1
  blitter 88DN.11
  CIA (hardware) HW.appxF, HW.appxF.2
  CIA TREF.6.2
  compatibility LD.Preface
  copper (hardware) 88DN.8, HW.2
  image data LD.8.1
  memory type LD.15.2
  overview (hardware) HW.1.1
  part number AMT1.1.4
  pin allocations (hardware) HW.appxC
  register lists (hardware) HW.appxA
  register list IA.H
Chunk
  See also IFF
  iff definition IA.I
  iffparse.library 90DN.11
CIA
  See also 8520, Chip
  cia.resource AMT1.9.2, AMT1.9.4, LD.43.3
  drive timing AMT1.8.4
  floppy disk control (hardware) HW.8.3
  floppy disk DMA (hardware) HW.8.3
  joyport output AMT1.11.4
  low-level 89DN.22
  register map 89DN.22
  resource usage 89DN.20
  spec (hardware) HW.appxF
  timer.device LD.41.1
  UNIT_MICROHZ LD.41.2
  using timer AMT1.9.2
Circle
  drawing LD.23.4
Class
  boopsi 90DN.32
Clear
  memory initialization LD.15.2
  set raster (graphics) LD.23.4
Client
  NFS 90DN.20
  Novell 90DN.21
Clipboard.device
  See also Clipboard
Clipboard
  clipboard.device LD.32
  commands LD.32.2
  data (IFF) LD.32.3
  docs 88DN.15, AMT1.14.2
  examples (cbio.c) ND20.3.9
  examples (clip.c (1.3)) LDd.2.10
  examples (clip.c) ND20.3.9
  examples (clipboard io) AMT1.14.2
  examples (cliphandler.c) 88d.1.5
  examples (FTXT) LD.32.6
  iffparse.library 90d.2.4
  readme ND20.3.9
  standards 90DN.17
  support code LD.32.7
Clip
  user clip rect LD.26.5
CLI
  commands ADOS.1.2
  error codes and messages ADOS.1.5
  introduction ADOS.1.1
  keymapping AMT1.11.3
  preferences (2.0) 90DN.10
  preferences control (1.3) AMT1.10.2
  program environment ADOS.2.1
  structure (process) ADOS.3.3
  user docs (1.3) EN13.2
  window pointer AMT1.2.13
Clock
  A3000 local bus 90DN.1
  constant (PAL/NTSC) AMT1.1.4
  constants for audio AMT1.6.1
  expansion 88DN.24
  frequency (PAL/NTSC) AMT1.1.10
CLUT
  See also IFF
  IFF (color look-up table) AMT1.7.4
  specification 89DN.14
CMAP
  See also IFF
  introduction AMT1.7.1
Coercion
  screen (2.0) 90DN.29
Collision
  animation AMT1.5.6, LD.25.8
  gel to border AMT1.5.6
  gel to gel AMT1.5.6
  port names 88DN.6
  sprite (hardware) HW.7.3
  vsprite LD.25.4
Color
  A2410 hires card 90DN.35
  bob LD.25.7
  compatibility AMT1.1.10
  deep ILBMs AMT1.7.4
  draw pattern LD.23.4
  drawinfo specification (2.0) 90DN.29
  drawing pens LD.23.4
  dual playfield (low-level) HW.3.3
  fonts AMT1.5.2
  font 88DN.14
  in viewport LD.23.2
  look-up (CLUT) AMT1.7.4

playfield (low-level) HW.3.2
printer driver writing 88DN.21
printer LD.39.4
register summary (hardware) HW.3.8
set raster LD.23.4
sprite (hardware) HW.4.1
sprite register (hardware) HW.4.12
sprite LD.25.2
text LD.24.4
vsprite LD.25.4
Command key
menu LD.5.1
style LD.11.4
Command line
processing (2.0) 90DN.31
Commands
CLI ADOS.1.2
Command
arexx 89DN.11
device (standard) LD.19.4
DOS command usage (1.3) 88DN.1
scsi 90DN.33
Comment
CLI ADOS.1.2
Commodities
examples (Blanker (2.0)) 90d.1.4
examples (blanker.c) ND20.3.2
examples (common files) ND20.3.2
examples (IHelp (2.0)) 90d.1.4
examples (ihelp.c) ND20.3.2
examples (LeftyMouse (2.0)) 90d.1.4
examples (leftymouse.c) ND20.3.2
examples (MBA (2.0)) 90d.1.4
examples (mba.c) ND20.3.2
examples (NoCapsLock (2.0)) 90d.1.4
examples (nocapslock.c) ND20.3.2
examples (preliminary 2.0) 89d.1.2
examples (skeleton) 90DN.25
examples (template) ND20.3.2
overview (1.4) 89DN.18
overview (2.0) 90DN.25
Compatibility
See also Debugging
articles AMT1.1
assembler code 88DN.20
DOS (2.0) 90DN.31
exception vectors AMT1.3.1
general LD.appxA
guidelines LD.Preface
international 89DN.21, AMT1.11.1, AMT1.4.1, LD.33.11
issues (1.4) 89DN.22
low level HW.1.3
model AMT1.1.3, LD.appxA
OS version LD.appxA
overscan 88DN.7
overview (2.0) 90DN.16
PAL and NTSC AMT1.1.4
preferences (1.4) 89DN.15
problems AMT1.1.11, LD.appxA
processor independent waits AMT1.9.4
processor 88DN.27, LD.appxA
programming rules IA.Preface, LD.appxA
testing LD.appxA.2
troubleshooting LD.appxA
warnings AMT1.1.1, LD.Preface
COMPLEMENT
draw mode LD.23.4
Component
animation LD.25.10
Compression
ANIM 88DN.3
IFF 88DN.3, AMT1.7.1
CON:
See also Console
control sequence ADOS.2.5
DOS windows ADOS.2.5
entering characters ADOS.2.5
examples (clicolors (packet)) AMT1.2.10
examples (set raw mode) AMT1.2.9
keymapping AMT1.11.3
packets AMT1.2.10
raw events ADOS.2.5, AMT1.2.9
raw mode AMT1.2.9
Configuration
expansion.library LD.27
Connectors
A1000 (hardware) TREF.1
A2000 (hardware) TREF.1
A500 (hardware) TREF.1
Console
See also CON:
control sequence LD.33.6
conunit LD.33.5
docs LD.33
examples (console.c (1.3)) LDd.2.9
examples (console.device) LD.33.8
examples (keymap source) LD.33.12
examples (set raw mode) AMT1.2.9
keymapping AMT1.11.3, LD.33.11
raw events ADOS.2.5
without window LD.33.10
Consonant
selection LD.37.5
Context
iffparse.library 90DN.11
Control characters
string gadgets (2.0) 90DN.29
Control sequence
CON: ADOS.2.5
console.device LD.33.6
console LD.33.1
printer.device LD.39.3
raw AMT1.2.9
Conunit
structure LD.33.5
Conversion
FFP math LD.28.5
Coordinates
border LD.8.1
Copper
changes (1.4) 89DN.6.2
control through intuition LD.10.5
hardware HW.2
in display LD.23.1
in view LD.23.2
instruction summary (hardware) HW.2.9
interrupt (hardware) HW.7.5
lists (hardware) HW.2.6
loops (hardware) HW.2.8
low level programming HW.2.1
programming 88DN.8
user copper lists LD.23.5
vsprites LD.25.5
Coprocessor slot
A2000 expansion 88DN.24
Copyright
software 89DN.9
Copy
arbitrary rectangle (blitter) HW.6.7
blitter 88DN.11
memory LD.15.4
rectangle LD.23.4
CPU
See also Processor
compatibility 88DN.27
identification 88DN.27
identifying AMT1.3.5
Crashes
causes AMT1.1.11, LD.appxA
debugging tools 90d.3.4
SRT tool (1.3) 88d.4.7
CreateExtIo
amiga.lib LD.appxB.2
CreatePort
amiga.lib LD.appxB.2
CreateProc
using AMT1.2.12
CreateTask
amiga.lib LD.appxB.2
CRNG
See also IFF
CTrac
CDTV emulation 90DN.12
Currency
international 90DN.26
Cursor key
style LD.11.4
Cursor
console position LD.33.8
position LD.24.2
Custom chip
See also Chip
overview (hardware) HW.1.1
pin allocations (hardware) HW.appxC
register lists (hardware) HW.appxA
Custom gadget
hooks 90DN.32
implementation 90DN.32
CX
commodities exchange (1.4) 89DN.18
Cycle
gadtools (2.0) 90DN.30
Damage list
layers LD.26.5
Database
AmigaVision 90DN.22
Data
file block ADOS.3.1
Date
international 90DN.26
dbf
amiga.lib LD.appxB.2
Ddebug.lib
electronic (1.3) ND13.3.7
Dead key
conversion AMT1.11.3
DeadKeyConvert LD.9.4
double-dead keys LD.33.11
keymap LD.33.11
RawKeyConvert LD.33.10
using AMT1.11.1
Dead-end
alert LD.6.2
Debug.lib
autodocs (1.3) IA.F
electronic (1.3) ND13.3.7
linker lib LD.appxB.3
Debugging
See also Compatibility, Tools (debugging)
assembler 88DN.20
audio AMT1.6.5
debug.lib LD.appxB.3
examples (in assembler) 90d.3.3
examples (in C) 90d.3.3
examples (kprintf and dprintf) 90d.3.3
general 90DN.12
linker libs 90d.3.10
macros 90d.3.3
MMU tools 90d.3.4, 90DN.12
parallel 90d.3.10
postscript printers AMT1.10.5
printer drivers AMT1.10.6
printer LD.39.4
remote 90DN.12
ROM Wack LD.22
serial 90d.3.10, LD.appxB.3
techniques LD.appxA.1
tools (1988) SWT.1
tools (1989) 89d.2.1
tools (1990) 90d.3.1, 90DN.12
troubleshooting guide AMT1.1.11, LD.appxA
Wack debugger SWT.2
Default
fonts 88DN.10
Delay
Delay() ADOS.2.2
low level CIA timer AMT1.9.2, AMT1.9.4
timer.device LD.41.3
DeleteExtIO
amiga.lib LD.appxB.2
DeletePort
amiga.lib LD.appxB.2
DeleteTask
amiga.lib LD.appxB.2
Denise
See also Chip
pin assignment (hardware) HW.appxC
Density

printer LD.39.4
Developer support
CATS IA.Preface, LD.Preface
Device list
location AMT1.2.3
DeviceProc
message port AMT1.2.3
Device
Amix 89DN.19
assembler IO AMT1.3.2
autodocs (1.3) IA.B
command index (1.3) IA.J
commands LD.19.4
compatibility AMT1.1.9
debugging tools 90d.3.6
examples (device source) IA.G
examples (source) IA.G
exec LD.19
expunging 88d.4.15, 89d.2.11
io LD.19
library vs. device LD.14.4
logical (DOS) ADOS.1.1
manual LD.Preface
quick io LD.19.5
sample device (1989) 89d.2.3
sample device (1990) LDd.2.20
DevInfo
structure definition AMT1.2.3
Diagrams
See also Mechanicals, Schematics
A1000 layout A1000.appxA
Diag
expansion events LD.27.3
Digital
to analog (audio) AMT1.7.3
Directory
current ADOS.1.1
file block ADOS.3.1
font contents LD.24.9
scanning (2.0) 90DN.31
using ADOS.1.1
Disable
gadget (by image) AMT1.4.2
gadget LD.4.3
interrupt LD.21.4
menu LD.5.1
tasks LD.17.6
Disassembler
Wack SWT.2
Disk font
diskfont.library LD.24.7
Disk object
See also Icon
manipulate LD.30.2
structure LD.30.3
Diskfont.library
See also Font
compatibility (2.0) 90DN.16
examples (dfont2obj.c) 88d.1.5
DiskInfo
function call AMT1.2.3
Disk
block types LD.27.3
boot process LD.appxC
changes (diskchange) LD.42.10
compatibility AMT1.1.3
connector (hardware) TREF.1
connector specifications (hardware) HW.appxE
controller port (hardware) HW.8.2
copying LD.42.8
debugging tools 90d.3.8, 90DN.12
disk.resource AMT1.8.4, LD.43.2
DiskEd tool SWT.10
drive ID (hardware) TREF.1
drive ID A1000.1.2
examples (available disk devices) AMT1.1.9
floppy drive design AMT1.8.3
font LD.24.10
formatting LD.42.8
hard disk software installation AMT1.1.7
hardware interface HW.appxI
IDCMP LD.7.3
interrupt (hardware) HW.7.5
interrupts (hardware) HW.8.3
low level access AMT1.8.2, AMT1.8.3, AMT1.8.4, LD.42.11
low-level programming 88d.4.15, 89d.2.11
MFM encoding LD.appxC
PC board schematics A1000.appxA.3
physical layout LD.appxC
pin out (hardware) HW.appxE
piracy 88DN.31
registers (hardware) HW.8.3
root block AMT1.8.1
timing (low-level) AMT1.8.4
trackdisk layout AMT1.8.1
trackdisk.device LD.42
type LD.42.9
virus 88DN.31, 88DN.32
Display
3D AMT1.14.3
A2024 (1.4) 89DN.6.1
A2410 hires card 90DN.35
beep LD.10.7
bitmap LD.23.1
CAMG format (2.0) 90DN.11
CDTV 90DN.12, 90DN.8
database (1.4) 89DN.6
display clip (1.4) 89DN.6.2
display id (1.4) 89DN.6.1
double buffered LD.23.3
double-buffered (low-level) 88DN.8
dual playfield LD.23.3
extra halfbrite LD.23.3
HAM LD.23.3
hold and modify LD.23.3
low level programming HW.3
mode properties (1.4) 89DN.6.1
modes AMT1.5.5, LD.23.4
monitors AMT1.5.5
new look (2.0) 90DN.29
output hardware HW.8.7
overscan 88DN.7
pens LD.23.4
printing LD.39.3
priority (hardware) HW.7.2
productivity (1.4) 89DN.6.1
query modes (1.4) 89DN.6.1
resolutions LD.23.1
rethink/remake LD.10.5
routines LD.23.2
selectable PAL/NTSC (1.4) 89DN.6.1
sprite (hardware) HW.4.3
superhires (1.4) 89DN.6.1
text LD.24.2
vsprites LD.25.5
Dithering
CDTV 90DN.12
DLTA
See also IFF
ANIMbrush format 90DN.11
IFF 88DN.3
DMA
audio (hardware) HW.5.2
audio sample size AMT1.7.3
blitter (hardware) HW.6.3
blitter 88DN.11
control (hardware) HW.7.6
CPU compatibility 88DN.27
sprite (disable) AMT1.5.4
sprite (hardware) HW.4.3
sprite LD.25.2
DoCollision
using AMT1.5.6, LD.25.4
DoIO
function description LD.19.3
Dos.library
See also AmigaDOS, DOS
DOS
See also AmigaDOS, Dos.library
articles AMT1.2
block size AMT1.2.6
check for terminal ADOS.2.2
command usage (1.3) 88DN.1
compatibility (2.0) 90DN.16
developer docs ADOS.2
directory handling ADOS.2.2
disk handling ADOS.2.2
DiskInfo AMT1.2.3
dos node LD.27.3
dos.library ADOS.2.2
error handling ADOS.2.2
examples (available disk devices) AMT1.1.9
examples (doslist.c) 88d.1.2
examples (dos_calls.c (2.0)) 90d.2.1
examples (dos_calls.c) ND20.3.4
examples (exall.c (2.0)) 90d.2.1
examples (exall.c) ND20.3.4
examples (get stack size) AMT1.2.11
examples (handler) 88d.1.2
examples (loadseg.asm) 88d.3.4
examples (multi-process) AMT1.2.12
examples (packet sending) AMT1.2.3
examples (pattern.c) ND20.3.4
examples (port-handler.c) 88d.1.2
examples (readargs (2.0)) 90DN.31
examples (readargs.c (2.0)) 90d.2.1
examples (readargs.c) ND20.3.4
examples (trackcopy.c (packets) (1.3)) LDd.2.17
file handling ADOS.2.2
file system design 88DN.23.
Flush AMT1.2.3
functions (2.0) 90DN.31
handler design 88DN.23
infodata AMT1.2.6
introduction 89DN.1.2
loading code ADOS.2.2
manual ADOS.2
MoreCache AMT1.2.3
name space 88DN.6
overview (1.4) 89DN.17, 89DN.23
packets (1.2) AMT1.2.3
packets AMT1.2.9
path internals AMT1.2.3
paths AMT1.2.5
process handling ADOS.2.2
process startup ADOS.2.1
process structures ADOS.3.3
resident list AMT1.2.3
seglists ADOS.3.3
SetFileDate AMT1.2.3
SetRawMode AMT1.2.3
stack size AMT1.2.11
standards 90DN.17
stream handling (io) ADOS.2.2
structures (1.2) AMT1.2.3
structures ADOS.3.3
technical docs ADOS.3
tools SWT.10
user docs (1.3) EN13.2
user docs ADOS.1
which.c AMT1.2.5
Double buffered
animation LD.25.11
audio LD.31.2
display (copper) 88DN.8
display LD.23.3
examples (collide.c) AMT1.5.6
Double-menu
requester LD.6.1
Dpaint
and IFF AMT1.7.1
DPAN
See also IFF
ANIMbrush format 90DN.11
Drawer
workbench LD.30.1
Draw
circle LD.23.4
ellipse LD.23.4
fill LD.23.4
line LD.23.4
modes LD.23.4
pen size LD.23.4
pens LD.23.4
rectangle LD.23.4
routines LD.23.4
sequenced LD.25.10

Driver
  expansion LD.27.3
  printer LD.39.4
Drive
  compatibility AMT1.1.9
Dual playfield
  display mode LD.23.2
  display LD.23.3
  field priority LD.23.2
  in CAMG AMT1.7.5
  low level programming HW.3.3
  screen viewmode LD.2.5
Dump
  printer graphics dump (1.3) AMT1.10.3
ECS
  display modes (2.0) 90DN.11
  display AMT1.5.5
  support (1.4) 89DN.4
Editing
  string gadget (2.0) 90DN.29
Editor
  ed commands ADOS.1.3
  edit commands ADOS.1.4
  preferences (2.0) 90DN.10
EDIT
  tutorial ADOS.1.4
ED
  tutorial ADOS.1.3
EHB
  See also Extra halfbrite
Ellipse
  drawing LD.23.4
Enable
  gadget LD.4.3
  menu LD.5.1
  tasks LD.17.6
ENV:
  and preferences (1.4) 89DN.15
  preferences (2.0) 90DN.10
Environment
  preferences (2.0) 90DN.10
Error reports
  submission LD.Preface
Error
  codes (CLI) ADOS.1.5
  printer LD.39.3
  serial LD.40.8
Escape sequence
  console AMT1.2.9, LD.33.6
  printer.device LD.39.3
Ethernet
  card 90DN.4
  TCP/IP 90DN.20
Event processing
  modular approach AMT1.4.5
Examples (Amix)
  device driver 89DN.19
  par.c 89d.2.14
Examples (animation)
  anim.c (animob) (1.3) LDd.1.3
  animtools.c (gels) (1.3) LDd.1.3
  anim_ex.c (1.3) LDd.1.3
  bob.c (1.3) LDd.1.3
  bob LD.25.7
  collide.c AMT1.5.6
  complete LD.25.11
  ssprite.c (1.3) LDd.1.3
  vsprite.c (1.3) LDd.1.3
Examples (AppShell)
  projlist (2.0) 90d.2.5
  watchman.c (2.0) 90d.2.5
Examples (arexx)
  minrexx.zoo (1989) 89d.2.12
  simple tools 90DN.27
  simplerexx.c AMT1.14.5
Examples (asl.library)
  asltest.c ND20.3.5
Examples (assembler)
  blitter 88d.1.4
  debug 88d.1.4
  device IO AMT1.3.2
  general 88d.1.4
  intserver 88d.1.4
  Midi output AMT1.9.6
  open library IA.A
  Rstartup.asm 88d.3.3
  spp 88d.3.2
  subroutines 88d.1.4
  vertical blank interrupt 88DN.20
Examples (audio)
  atools.c 88d.2.1
  audio cutoff filter AMT1.6.4
  audio.device I/O AMT1.6.2
  audio1.c (1.3) LDd.2.11
  audio2.c (1.3) LDd.2.11
  audiotools3.c 88d.2.1
  audio 88DN.12
  convertsound.c AMT1.6.6
  debtones.asm AMT1.6.5
  double-buffered LD.31.2
  simple audio LD.31.2
  touch tone sound AMT1.6.3
Examples (autoboot)
  DemoBootRomCode 88d.3.4
  loadseg.asm 88d.3.4
Examples (autodoc)
  autodoc.doc ND20.3.18
  autodoc.sample ND20.3.18
  autodoc.style ND20.3.18
Examples (blitter)
  various 88d.2.4
Examples (C language)
  open library IA.A
  reentrant code 88DN.2
Examples (clipboard)
  cbio.c ND20.3.9
  clip.c (1.3) LDd.2.10
  clip.c ND20.3.9
  clipboard io AMT1.14.2
  cliphandler.c 88d.1.5
  FTXT LD.32.6
Examples (commodities)
  assorted (preliminary 2.0) 89d.1.2
  Blanker (2.0) 90d.1.4
  blanker.c ND20.3.2
  common files ND20.3.2
  IHelp (2.0) 90d.1.4
  ihelp.c ND20.3.2
  LeftyMouse (2.0) 90d.1.4
  leftymouse.c ND20.3.2
  MBA (2.0) 90d.1.4
  mba.c ND20.3.2
  NoCapsLock (2.0) 90d.1.4
  nocapslock.c ND20.3.2
  skeleton 90DN.25
  template ND20.3.2
Examples (console)
  clicolors (packet) AMT1.2.10
  console.c (1.3) LDd.2.9
  console.device LD.33.8
  keymap source LD.33.12
  set raw mode AMT1.2.9
Examples (debugging)
  debtones.asm ND20.3.12
  dtest.asm ND20.3.12
  in assembler 90d.3.3
  in C 90d.3.3
  kprintf and dprintf 90d.3.3
  ktest.asm ND20.3.12
  mydebug.h ND20.3.12
  RawPrintf.asm ND20.3.12
  testmydebug.c ND20.3.12
  togl_led.asm ND20.3.12
Examples (device)
  source IA.G
Examples (diskfont)
  dfont2obj.c 88d.1.5
  Examples (graphics) LDd.1.1
Examples (DOS packet)
  trackcopy.c (1.3) LDd.2.17
Examples (DOS)
  available disk devices AMT1.1.9
  doslist.c 88d.1.2
  dos_calls.c (2.0) 90d.2.1
  dos_calls.c ND20.3.4
  exall.c (2.0) 90d.2.1
  exall.c ND20.3.4
  get stack size AMT1.2.11
  handler 88d.1.2
  multi-process AMT1.2.12
  packet sending AMT1.2.3
  pattern.c ND20.3.4
  port-handler.c 88d.1.2
  readargs (2.0) 90DN.31
  readargs.c (2.0) 90d.2.1
  readargs.c ND20.3.4
Examples (exec)
  allocate.c (1.3) LDd.1.17
  allocentry.c (1.3) LDd.1.17
  cold reboot AMT1.3.3
  complete message LD.18.3
  cpu identification AMT1.3.5
  cycvb.c (vb int) IFFd.6
  device creation (1.3) LDd.2.20
  device io LD.19.6
  expunge device AMT1.9.1
  io_example.c (1.3) LDd.1.18
  liblist.c (1.3) LDd.1.15
  library creation (1.3) LDd.2.6
  library creation AMT1.3.4
  lists.c (1.3) LDd.1.16
  list LD.16.6
  modular event handling AMT1.4.5
  msgport creation LD.18.2
  open library IA.A
  port.c (1.3) LDd.1.14
  put message LD.18.3
  RBFHandler (serial interrupt) LD.21.2
  rbfhandler_c.c (1.3) LDd.1.19
  sample device (1.3) LDd.2.20
  sample device 88d.4.3, 89d.2.3
  sample library (1.3) LDd.2.6
  sample library 88d.4.3, 89d.2.3
  scan list LD.16.6
  semaphore LD.20
  showcpu 88d.4.3
  simpletask.c (1.3) LDd.1.15
  task creation LD.17.3
  task list LD.17.6
  tasklist.c (1.3) LDd.1.15
  timersoftint.c (1.3) LDd.1.19
  TimerSoftInt LD.21.3
  trap handler LD.17.8
  trap_c.c (1.3) LDd.1.15
  vector base register) AMT1.3.1
  Vertb (interrupt) LD.21.2
  vertb_c.c (1.3) LDd.1.19
  wait LD.18.3
  wedge.c 88d.1.1
Examples (expansion)
  autoboot fragment LD.27.3
  autoboot.asm (1.3) LDd.2.4
  findboards.c (1.3) LDd.2.4
  query expansion AMT1.12.4
Examples (gadtools)
  gadget1.c (2.0) 90d.2.3
  gadget1.c ND20.3.6
  gadget2.c (2.0) 90d.2.3
  gadget2.c ND20.3.6
  gadtools1.c (2.0) 90d.2.3
  gadtools1.c ND20.3.6
  menu1.c (2.0) 90d.2.3
  menu1.c ND20.3.6
Examples (gameport)
  joy.c (1.3) LDd.2.8
  joystick LD.34.3
Examples (graphics)
  anim.c (animob) (1.3) LDd.1.3
  animation LD.25.11
  animtools.c (gels) (1.3) LDd.1.3
  anim_ex.c (1.3) LDd.1.3
  area fill LD.23.4
  areinfoexample.c (1.3) LDd.1.2
  available fonts LD.24.8
  bob.c (1.3) LDd.1.3
  bob LD.25.7
  boxfill.c (1.3) LDd.1.2
  collide.c AMT1.5.6

demo (1.4) 89DN.4.2
display mode id (2.0) AMT1.7.5
dual-playfield LD.23.6
dualpf.c (1.3) LDd.1.2
fontparade.c (1.3) LDd.1.1
genlock (preliminary 2.0) 89d.1.1
genlock_example.c (preliminary 2.0) 89d.1.1
hamexample.c (1.3) LDd.1.2
hold and modify LD.23.6
low-level view LD.23.2
primitives LDd.1.2
readpixelarray.c 88d.1.3
rectangle fill LD.23.4
rgbboxes.c (1.3) LDd.1.2
sample font LD.24.11
scale_rastport.c 88d.1.3
showdefaultfont.c (1.3) LDd.1.1
showdrawmodes.c (1.3) LDd.1.1
ShowDrawModes LD.24.5
showopenfont.c (1.3) LDd.1.1
ShowOpenFont LD.24.3
sprite switch AMT1.5.4
sprite LD.25.2
ssprite.c (1.3) LDd.1.3
suits8.asm (font) (1.3) LDd.1.1
tutorial.c (preliminary 2.0) 89d.1.1
ucopperexample.c (1.3) LDd.1.2
user copper lists LD.23.5
viewport (preliminary 2.0) 89d.1.1
viewportdemo (preliminary 2.0) 89d.1.1
vsprite.c (1.3) LDd.1.3
vsprite LD.25.6
Examples (hardware)
basic playfield HW.3.2
blitter clear HW.6.17
blitter line HW.6.17
blitter rotate HW.6.17
hold and modify HW.3.7
peripheral design 89DN.8
sprite HW.4.3
Examples (icon)
See also Examples (workbench)
icon demo LD.30.6
rkm_icon_example.c (1.3) LDd.2.3
Examples (IEEE)
See also Examples (math)
Examples (IFF)
See also Tools (IFF)
8SVX player AMT1.7.3
ANIM 88d.2.2, IFFd.6
code modules IFFd.2
Display (1.2) IFFd.6
display mode id (2.0) AMT1.7.5
FTXT clipboard LD.32.6
iff code IA.I
iffparse.library 89DN.14
PGTB 88d.2.2, IFFd.6
ScreenSave (1.2) IFFd.6
ShowAnim 88d.2.2, IFFd.6
SMUS player AMT1.7.2
Examples (iffparse.library)
looki.c 90d.2.4, ND20.3.7
sift.c 90d.2.4, ND20.3.7
simple.c ND20.3.7
tiff.c 90d.2.4, ND20.3.7
tiff2.c ND20.3.7
tree.c 90d.2.4, ND20.3.7
wiff.c 90d.2.4, ND20.3,7
wiff2.c ND20.3.7
Examples (input)
idio.c 88d.1.8
iefa.asm 88d.1.8
inputhandler.a (1.3) LDd.2.16
inputmake.c (1.3) LDd.2.16
inputset.c (1.3) LDd.2.16
inputswap.c (1.3) LDd.2.16
joymouse.c 88d.1.8, AMT1.11.2
joystick port output AMT1.11.4
keymapping AMT1.11.3
propjoy.c 88d.1.8, AMT1.11.2
swap input LD.35.4
Examples (international)
compat.c 89d.2.15
strings.asm 88d.1.5
Examples (intuition)
2024.c (screens) ND20.3.3
alert (1.3) LDd.1.8
autorequester.c (1.3) LDd.1.8
autorequest LD.6.3
backdrop window LD.3.3
boopsi (2.0) 90d.1.5
boopsi demos ND20.3.3
borderless.c (1.3) LDd.1.6
buildreq 88d.1.7
cgmain.c ND20.3.3
closewindowsafely.c (1.3) LDd.1.11
CloseWindowSafely LD.7.4
custgad (2.0) 90d.1.5
cwb.c (screens) ND20.3.3
dblmenureq.c (1.3) LDd.1.8
dcop 88d.1.7
dead key convert LD.9.4
democlasslib.c ND20.3.3
demoframe.c ND20.3.3
demoimage.c ND20.3.3
demopubi.c ND20.3.3
demotextb.c ND20.3.3
dialgadget.c ND20.3.3
disabled render AMT1.4.2
display alert LD.6.3
displayalert.c (1.3) LDd.1.8
double-menu request LD.6.3
dual-playfield screen LD.2.5
dualpf.c (1.3) LDd.1.7
dvdemo 88d.1.7
EasyReq.c (2.0) 90d.1.5
ellipse.c ND20.3.3
emboxclass.c ND20.3.3
emboxpubcl.c ND20.3.3
event loop LD.1.4
ez.c ND20.3.3
font accomodation 90d.3.15, LD.5.2
frame1class.c ND20.3.3
full screen AMT1.4.1
gadgets.c (1.3) LDd.1.12
gadgets.c LD.4.5
greetings.c (1.3) LDd.1.7
hookface.asm ND20.3.3
IDCMP event loop LD.7.4
idcmpdemo.c (1.3) LDd.1.11
ihooks (preliminary 2.0) 89d.1.3
image data LD.8.1
images.c ND20.3.3
invisible pointer LD.3.3
limit keystroke repeat AMT1.4.4
lines.c (1.3) LDd.1.6
lo-res screen LD.2.5
locklist.c (screens) ND20.3.3
memory allocation LD.10.3
menus.c (1.3) LDd.1.12
menu LD.5.2
modular event handling AMT1.4.5
mouse and keyboard LD.9.4
mousekeys.c (1.3) LDd.1.10
mutual exclude AMT1.4.3
myclass.library ND20.3.3
myclassinit.c ND20.3.3
myclasslib.asm ND20.3.3
mygroupgclass.c ND20.3.3
mymodelclass.c ND20.3.3
open library IA.A
oscan 88d.1.7
overscan (1.3) 88d.1.7
pointer to image AMT1.5.1
prop gadget LD.4.3
pub.c (screens) ND20.3.3
pubi.c ND20.3.3
pubsc.c (preliminary 2.0) 89d.1.3
pubsc.c ND20.3.3
remembertest.c (1.3) LDd.1.9
renderinfo.c 90d.3.15
scdemo (2.0) 90d.1.5
scdemo.c (preliminary 2.0) 89d.1.3
scdemo.c ND20.3.3
sdm.c (screens) ND20.3.3
strdemo (2.0) 90d.1.5
strgad.c ND20.3.3
strhooks.c ND20.3.3
superbitmap LD.3.3
tagdebug.c ND20.3.3
textbclass.c ND20.3.3
two window LD.3.3
twowindows.c (1.3) LDd.1.6
ucop 88d.1.7
visitor.c (screens) ND20.3.3
vm.c (screens) ND20.3.3
Examples (joystick)
See also Examples (input)
Examples (keyboard)
keyhandler.a (1.3) LDd.2.15
keyinput.c (1.3) LDd.2.15
keymatrix.c (1.3) LDd.2.15
keyreset.c (1.3) LDd.2.15
matrix LD.36.2
read event LD.36.3
reset handler LD.36.2
Examples (keymap)
example source LD.33.12
keymap.c 88d.1.5
Examples (layers)
clip window LD.26.5
layer operations LD.26.5
layer.c (1.3) LDd.2.5
Examples (library)
source IA.G
Examples (locale.library)
catalogue.sample (2.0) 90d.2.2
CatGen (2.0) 90d.2.2
getmsg.c (2.0) 90d.2.2
locale_lib.asm (2.0) 90d.2.2
Examples (math)
div0 (IEEE) 88d.3.2
div0.c (1.3 IEEE) ND13.4.3
fast fourier transforms AMT1.13.2
ffpconv.c (1.3) LDd.1.4
ffuz.c (1.3 IEEE) ND13.4.3
Examples (misc.resource)
flushdevice.c (1.3) LDd.2.22
serial and parallel AMT1.9.1
Examples (mouse)
See also Examples (input)
Examples (narrator)
narrator.c (1.3) LDd.2.14
speech LD.37.7
Examples (parallel)
parallel.c (1.3) LDd.2.13
parallel2.c (1.3) LDd.2.13
parallel3.c (1.3) LDd.2.13
Examples (ports)
joyport_output 88d.4.3
Examples (potgo.resource)
read9n5.c (1.3) LDd.2.23
Examples (printer)
colormaster (1.3) ND13.4.8
colormaster2 (1.3) ND13.4.8
colormaster2 88d.3.1
colormaster 88d.3.1
epsonq (1.3) ND13.4.8
epsonq 88d.3.1, LD.39.9
epsonx (1.3) LDd.2.12, ND13.4.8
epsonx LD.39.9
graphic dump LD.39.3
hp (1.3) LDd.2.12, ND13.4.8
hp_laserjet LD.39.9
hp 88d.3.1
okimate20 (1.3) ND13.4.8
okimate20 88d.3.1
simple print functions AMT1.10.7
wdump.c (1.3) LDd.2.12
xerox_4020 (1.3) LDd.2.12, ND13.4.8
xerox_4020 88d.3.1, LD.39.9
Examples (SCSI)
modesense.c ND20.3.8
scsi modesense.c 90d.1.1
Examples (serial)
serial.c (1.3) LDd.2.19
serial2.c (1.3) LDd.2.19
Examples (startup)

reentrant code 88DN.2
startup code LD.30.6
workbench startup code LD.30.6
Examples (timer)
8520 timing AMT1.9.4
getsystime.c (1.3) LDd.2.18
Midi output AMT1.9.6
software interrupt AMT1.9.5
stopwatch AMT1.9.7
timemath.c (1.3) LDd.2.18
timersoftint.c (1.3) LDd.1.19
timertest.c (1.3) LDd.2.18
timer LD.41.8
using cia timer AMT1.9.2
Examples (trackdisk)
low-level control AMT1.8.2
track copy LD.42.14
trackcopy.c (1.3) LDd.2.17
Examples (Unix)
See also Examples (Amix)
Examples (wack)
exec.macros ND20.3.20
graphics.macros ND20.3.20
intuition.macros ND20.3.20
wack.macros ND20.3.20
Examples (wedge)
dos wedges ND20.3.14
exec wedges ND20.3.14
icon wedges ND20.3.14
intuition wedges ND20.3.14
memory wedges ND20.3.14
Examples (workbench)
appicon.c (2.0) 90d.1.3
appicon.c (preliminary 2.0) 89d.2.2
AppIcon.c ND20.3.1
AppIcon.Image ND20.3.1
appmenu.c (2.0) 90d.1.3
AppMenu.c ND20.3.1
appmenuitem.c (preliminary 2.0) 89d.2.2
appwindow.c (2.0) 90d.1.3
appwindow.c (preliminary 2.0) 89d.2.2
AppWindow.c ND20.3.1
CLI and WB startup code (1.3) LDd.2.2
prargs.c (1.3) LDd.2.3
rkm_icon_example.c (1.3) LDd.2.3
startup code LD.30.6
Examples
See also Examples (subject), Subject, Tools
compiling c language LD.Preface
simple c language 88d.2.3
startup code 88d.3.3
Exception
processor LD.17.7
task stack LD.17.3
vectors AMT1.3.1
Exclude
mutual exclusion of gadgets AMT1.4.3
Executable file
layout ADOS.3.2
Executable
file layout ADOS.3.2
Exec
articles AMT1.3
compatibility (2.0) 90DN.16
debugging tools 90d.3.7
devices LD.19
examples (allocate.c (1.3)) LDd.1.17
examples (allocentry.c (1.3)) LDd.1.17
examples (cold reboot) AMT1.3.3
examples (complete message) LD.18.3
examples (cpu identification) AMT1.3.5
examples (device creation (1.3)) LDd.2.20
examples (device io) LD.19.6
examples (expunge device) AMT1.9.1
examples (io_example.c (1.3)) LDd.1.18
examples (liblist.c (1.3)) LDd.1.15
examples (library creation (1.3)) LDd.2.6
examples (library creation) AMT1.3.4
examples (limpletask.c (1.3)) LDd.1.15
examples (list) LD.16.6
examples (lists.c (1.3)) LDd.1.16
examples (msgport creation) LD.18.2
examples (open library) IA.A
examples (port.c (1.3)) LDd.1.14
examples (put message) LD.18.3
examples (RBFHandler (serial interrupt)) LD.21.2
examples (rbfhandler_c.c (1.3)) LDd.1.19
examples (sample device (1.3)) LDd.2.20
examples (sample device) 88d.4.3, 89d.2.3
examples (sample library (1.3)) LDd.2.6
examples (sample library) 88d.4.3, 89d.2.3
examples (scan list) LD.16.6
examples (semaphore) LD.20
examples (showcpu) 88d.4.3
examples (task creation) LD.17.3
examples (task list) LD.17.6
examples (tasklist.c (1.3)) LDd.1.15
examples (TimerSoftInt) LD.21.3
examples (timersoftint.c (1.3)) LDd.1.19
examples (trap handler) LD.17.8
examples (trap.c (1.3)) LDd.1.15
examples (vector base register) AMT1.3.1
examples (Vertb (interrupt)) LD.21.2
examples (vertb_c.c (1.3)) LDd.1.19
examples (wait) LD.18.3
examples (wedge.c) 88d.1.1
exceptions LD.17.7
exec.library LD.14
input and output LD.19
interrupts LD.21
introduction (io) 89DN.1.1
io request structure LD.19.2
io standard request structure LD.19.2
libraries LD.14
library creation AMT1.3.4
library vs. device LD.14.4
list introduction 89DN.1.1
lists LD.16
manual LD.14
memory allocation LD.15
messages LD.18.3
name space 88DN.6
overview (1.4) 89DN.20
overview (2.0) 90DN.34
ports LD.18.2
ROM Wack LD.22
semaphores LD.20
support functions LD.appxB.2
tasks LD.17
traps LD.17.8
Expansion
See also Autoboot, Autoconfig
100 Pin slot TREF.3.4
68030 compatibility AMT1.12.5
86 Pin slot TREF.3.4
A1000 A1000.2.1
A3000 90DN.1
A500 88DN.33
articles AMT1.12
autoboot boards 88DN.25
autoboot AMT1.12.3
board design 88DN.24, TREF.3.1
board ID information A1000.2.3, TREF.3.1
byte-wide roms AMT1.12.1
drivers LD.27.3, TREF.3.2
exampels (card) TREF.appxA.1
examples (autoboot fragment) LD.27.3
examples (autoboot.asm (1.3)) LDd.2.4
examples (backplane) TREF.appxA.1
examples (findboards.asm (1.3)) LDd.2.4
examples (query expansion) AMT1.12.4
expansion.library LD.27, TREF.3.3
hardware TREF.3.1
IO space timing AMT1.12.2
manufacturer number AMT1.12.4
PAL equations TREF.3.1
pin edge (A1000) HW.appxE
pin edge (hardware) HW.appxE
sequence LD.27.2
timing (hardware) TREF.3.1
video slot TREF.3.4
video 90DN.2
Export
issues 89DN.26
Expunge
device AMT1.9.1
Extended
TextFont structure definition AMT1.5.2
text LD.24.6
Extra halfbrite
display mode LD.23.1, LD.23.2
display LD.23.3
in CAMG AMT1.7.5
in IFF AMT1.7.1
screen viewmode LD.2.5
Extras
user docs (1.3) EN13.6
Fast File System
See also FFS, File System
bitmap block AMT1.2.4
boot block AMT1.2.4
data block AMT1.2.4
design AMT1.2.1
file header block AMT1.2.4
hashing AMT1.2.4
low level differences AMT1.2.4
root block AMT1.2.4
user directory block AMT1.2.4
user docs (1.3) EN13.appxA
Fast Floating Point
See also Math
Fast Fourier Transform
article AMT1.13.2
FastRand
amiga.lib LD.appxB.2
Fast
memory type LD.15.2
Fat agnus
pin assignment (hardware) HW.appxC
fclose
amiga.lib LD.appxB.2
Fd
electronic (1.3) ND13.3.7
function descriptions (1.3) IA.H
iffparse_lib.fd ND20.3.7
FFP
See also Math
math libraries LD.28.2
FFS
See also Fast File System, File System
design AMT1.2.1
low level differences AMT1.2.4
user docs (1.3) EN13.appxA
fgetc
amiga.lib LD.appxB.2
File handle
DOS ADOS.3.3
File requester
See also Asl.library
File system
alternate LD.27.3
block size AMT1.2.6
design tips 88DN.23
design 88DN.23
enhancements (1.4) 89DN.17.2
fast file system AMT1.2.1
Fast File System AMT1.2.4
FFS AMT1.2.1, AMT1.2.4
hardblocks LD.27.3
layout ADOS.3.1
rigid disk block 90DN.33
File
See also File System
executable file layout ADOS.3.2
handles (conversion (2.0)) 90DN.31
hunks ADOS.3.2
layout and blocks ADOS.3.1
naming ADOS.1.1
object file layout ADOS.3.2
opening ADOS.2.2
Fill
area LD.23.4
blitter (hardware) HW.6.8
blitter 88DN.11
graphics information LD.23.4
pattern LD.23.4
set raster LD.23.4
Filter

audio cutoff AMT1.6.4
audio low-pass (hardware) HW.5.4
Find
port LD.18.2
Flags
font LD.24.11
gadget LD.4.4
Floating point
formats AMT1.13.1
Flood fill
See also Fill
Floppy disk
trackdisk.device LD.42
Flush
function call AMT1.2.3
Font requester
See also Asl.library
Font
See also Diskfont.library
Amiga 88DN.14
application 88DN.10
available LD.24.8
character data LD.24.11
ColorText AMT1.5.2
color 88DN.14
compatibility AMT1.1.9
defaults 88DN.10
directory contents LD.24.9
disk LD.24.7
examples (creation) LDd.1.1
examples (fontparade) LDd.1.1
examples (sample font) LD.24.11
examples (showdefaultfont) LDd.1.1
examples (showdrawmodes) LDd.1.1
examples (showopenfont) LDd.1.1
output LD.24
run-time accomodation 90DN.37
sample source code LD.24.11
scaling 88DN.14, 90DN.15
sizes 90DN.37
size LD.24.2
string gadget (2.0) 90DN.29
structure LD.24.11
system use 88DN.10
text layout 88DN.10
Forbid
tasks LD.17.6
Format
FFP math LD.28.2
IEEE math LD.28.6
FORM
See also IFF
clipboard AMT1.14.2
iff definition IA.I
fpa
amiga.lib LD.appxB.2
fpbcd
amiga.lib LD.appxB.2
fprintf
amiga.lib LD.appxB.2
fputc
amiga.lib LD.appxB.2
fputs
amiga.lib LD.appxB.2
Free
memory (automatic) LD.15.6
memory LD.15.2
FTXT
See also IFF
clipboard AMT1.14.2, LD.32.6
IFF spec IFF.3.2
Full size
window LD.11.6
Function
autodocs IA.A
device (standard) LD.19.4
index (1.3) IA.J
library base offsets (1.3) IA.H
Gadget
boolean LD.4.3
Checkbox (2.0) 90DN.30
combinations (1.3) LD.4.3
custom (1.4) 89DN.12.2
custom gadget hooks 90DN.32
custom 90DN.32
Cycle (2.0) 90DN.30
disabled (alternate image) AMT1.4.2
enable/disable LD.4.3
examples (border) LD.4.5
examples (complete) LD.4.5
examples (custom gadget (2.0)) 90d.1.5
examples (gadgets.c) LDd.1.12
examples (image) LD.4.5
examples (intuitext) LD.4.5
functions LD.4.4
gadget structure LD.4.4
gadtools (2.0) 90DN.30
gadtools (readme) ND20.3.6
highlight LD.4.3
in requesters LD.6.1
in window border LD.4.3
Integer (2.0) 90DN.30
integer LD.4.3
intuition LD.4
Listview (2.0) 90DN.30
modify prop LD.4.4
Mutual exclude (2.0) 90DN.30
mutual exclude AMT1.4.3, LD.4.3
newmodifyprop changes (2.0) 90DN.29
Numeric display (2.0) 90DN.30
Palette (2.0) 90DN.30
position LD.4.3
proportional LD.4.3
refresh LD.4.3
render (border) LD.4.3
render (image) LD.4.3
Scroller (2.0) 90DN.30
selection LD.4.3
Slider (2.0) 90DN.30
specialinfo LD.4.4
String (2.0) 90DN.30
string changes (2.0) 90DN.29
string gadget LD.4.3
style LD.11.2
system LD.4.2
Text display (2.0) 90DN.30
type LD.4.1
Gadtools
examples (gadget1.c (2.0)) 90d.2.3
examples (gadget1.c) ND20.3.6
examples (gadget2.c (2.0)) 90d.2.3
examples (gadget2.c) ND20.3.6
examples (gadtools1.c (2.0)) 90d.2.3
examples (gadtools1.c) ND20.3.6
examples (menu1.c (2.0)) 90d.2.3
examples (menu1.c) ND20.3.6
in AppShell 90DN.28
overview (2.0) 90DN.30
readme ND20.3.6
Game controller
specifications (hardware) HW.appxE
Gameport
examples (joy.c (1.3)) LDd.2.8
examples (joystick) LD.34.3
gameport.device LD.34
input AMT1.11.2
resource LD.43.5
Game
vendor advice 88d.4.15, 89d.2.11
Gary
See also Chip
changes 88DN.24
Gel
animation objects AMT1.5.6
bob LD.25.7
collision LD.25.8
pointer to LD.23.4
system setup LD.25.1
vsprite LD.25.4
Genlock
ECS extensions (1.4) 89DN.4.1
examples (preliminary 2.0) 89d.1.1
Interface guidelines AMT1.12.6
with ECS (1.4) 89DN.6.2
getchar
amiga.lib LD.appxB.2
Gimmezerozero
window type LD.3.1
Global vector
definition ADOS.3.3
GlobVec
fast file system AMT1.2.1
Graphics card
See also Video slot
Graphics
A2410 hires card 90DN.35
Amix 89DN.16
animation LD.25.1
articles AMT1.5
bobs LD.25.7
compatibility (2.0) 90DN.16
drawing LD.23.4
ECS support (1.4) 89DN.4
examples (anim.c (1.3)) LDd.1.3
examples (animation) LD.25.11
examples (animtools.c (1.3)) LDd.1.3
examples (anim_ex.c (1.3)) LDd.1.3
examples (area fill) LD.23.4
examples (areainfoexample.c (1.3)) LDd.1.2
examples (available fonts) LD.24.8
examples (bob.c (1.3)) LDd.1.3
examples (boxfill.c (1.3)) LDd.1.2
examples (collide.c) AMT1.5.6
examples (demo (1.4)) 89DN.4.2
examples (display mode id (2.0)) AMT1.7.5
examples (dual-playfield) LD.23.6
examples (dualpf.c (1.3)) LDd.1.2
examples (fontparade.c (1.3)) LDd.1.1
examples (genlock (prelim 2.0)) 89d.1.1
examples (genlock_example.c (prelim 2.0)) 89d.1.1
examples (hamexample.c (1.3)) LDd.1.2
examples (hold and modify) LD.23.6
examples (low-level view) LD.23.2
examples (primitives) LDd.1.2
examples (readpixelarray.c) 88d.1.3
examples (rectangle fill) LD.23.4
examples (rgbboxes.c (1.3)) LDd.1.2
examples (sample font) LD.24.11
examples (scale_rastport.c) 88d.1.3
examples (showdefaultfont.c (1.3)) LDd.1.1
examples (ShowDrawModes) LD.24.5
examples (showdrawmodes.c (1.3)) LDd.1.1
examples (ShowOpenFont) LD.24.3
examples (showopenfont.c (1.3)) LDd.1.1
examples (sprite switch) AMT1.5.4
examples (sprite) LD.25.2
examples (ssprite.c (1.3)) LDd.1.3
examples (suits8.asm (font 1.3)) LDd.1.1
examples (tutorial.c (prelim 2.0)) 89d.1.1
examples (ucopperexample.c (1.3)) LDd.1.2
examples (user copper lists) LD.23.5
examples (viewport (prelim 2.0)) 89d.1.1
examples (viewportdemo (prelim 2.0)) 89d.1.1
examples (vsprite) LD.25.6
examples (vsprite.c (1.3)) LDd.1.3
fonts LD.24
graphics.library LD.23
hires card 89DN.10
introduction 89DN.1.2
intuition functions LD.8.1
intuition LD.8.1
low level programming HW.3
manual LD.23
overview (1.4) 89DN.6.2
primitives LD.23
printer driver writing 88DN.21, AMT1.10.6
printer driver LD.39.4
printer dump (1.3) 88DN.17
printer preferences AMT1.10.2
printing LD.39.3
sprites AMT1.5.1, LD.25.1
text LD.24
Unix 89DN.16
viewports LD.23.2
vsprite LD.25.4
Guidelines
programming LD.Preface
Guru

See also Alert, Software error
68000 exception numbers LD.17.8
Amiga exception numbers (exec/alerts.h) IA.D
causes AMT1.1.11, LD.appxA
HAM
See also Hold and modify
display LD.23.3
low level HW.3.7
screen viewmode LD.2.5
Handler
dos.library 88DN.23
input LD.35
interrupt LD.21.2
port-handler.c 88d.1.2
register usage LD.21.2
trap LD.17.8
Handle
file (DOS) ADOS.3.3
iffparse.library 90DN.11
Hardblocks
See also Rigid disk block
specification LD.27.3
Harddisk
autoboot 90DN.33
harddisk include files 89d.2.3
include files (1.3.1) 88d.4.4, 89d.2.3
installing 1.3 AMT1.2.7, AMT1.2.8
rigid disk block 90DN.33
software installation AMT1.1.7
Hardware
2090 TREF.5
68030 compatibility AMT1.12.5
8520 TREF.6.2
A1000 expansion A1000.2.1
A1000 A1000.1.2
A2000 motherboards AMT1.14.4
A3000 architecture 90DN.36
A3000 expansion 90DN.1, 90DN.2
A500 interfacing 88DN.33
advanced architectures 89DN.13
Agnus TREF.6.1
audio cutoff filter AMT1.6.4
audio HW.5
autoboot boards 88DN.25
autoconfig HW.appxG
blitter 88DN.11, AMT1.5.3, HW.6
block diagrams TREF.2
chip pin allocations HW.appxC
CIA HW.appxF.2, TREF.6.2
copper 88DN.8, HW.2
diagrams TREF.appxA
differences TREF.1
disk interface (hardware) HW.8.2
disk interface HW.appxI
display output HW.8.7
examples (basic playfield) HW.3.2
examples (blitter clear) HW.6.17
examples (blitter line) HW.6.17
examples (blitter rotate) HW.6.17
examples (hold and modify) HW.3.7
examples (peripheral design) 89DN.8
examples (sprite) HW.4.3
expansion 88DN.24
floppy access AMT1.8.3
harddisk TREF.5
hires color graphics card 89DN.10
interfaces HW.8, HW.appxE
interrupt register HW.appxF.6
interrupt HW.7.5
jumpers TREF.7.4
keyboard HW.8.4, HW.appxH
low level input AMT1.11.2
manual HW.1.1
new for 1989 89DN.13
new products 1990 90DN.19
overview HW.1.1
PAL equations TREF.7.3
parallel HW.8.5
peripheral design 89DN.8
playfield (hardware) HW.3
power budgets TREF.7.2
programming HW.1.3
realtime clock TREF.7.1
register include file HW.appxJ
register summary HW.appxA
schematics TREF.appxB
serial HW.8.6, HW.appxF.5, LD.40.10
sprite HW.4
system control HW.7
system memory map HW.appxD
takeover 88DN.9
timer HW.appxF
video board 90DN.2
Hashing
algorithm (FFS) AMT1.2.4
files (trackdisk) AMT1.8.1
Header
file block ADOS.3.1
HEAD
See also IFF
IFF spec IFF.5.4, IFFd.7
Help key
style LD.11.4
Help
system standards 90DN.28
Hex codes
keyboard matrix LD.36.2
High resolution
display mode LD.23.1, LD.23.2
High speed
serial LD.40.7
Highlight
gadget LD.4.3
Hires
in CAMG AMT1.7.5
playfield (low-level) HW.3.2
screen viewmode LD.2.5
Hit mask
using AMT1.5.6
Hold and modify
See also HAM
display mode LD.23.1, LD.23.2
display LD.23.3
in CAMG AMT1.7.5
in IFF AMT1.7.1
low level HW.3.7
Hooks
backfill (2.0) 90DN.29
callback (2.0) 90DN.29
custom gadget 90DN.32
string gadget editing (2.0) 90DN.29
Hot key
See also Commodities
introduction 90DN.25
Hot link
notification (2.0) 90DN.31
Hunk
file layout ADOS.3.2
I/O
audio.device AMT1.6.2
buffered (2.0) 90DN.31
CDTV 90DN.8
console LD.33.1
controller port (hardware) HW.8.2
DOS file handling ADOS.2.2
exec device LD.19
IDCMP flags LD.7.3
IDCMP LD.7
introduction (exec) 89DN.1.1
intuimessage LD.7.3
intuition overview LD.7.1
joystick port output AMT1.11.4
modify IDCMP LD.7.3
preparation LD.19.5
quick LD.19.5
request LD.19.2
Icon
appicon description (2.0) 90DN.24
examples (icon demo) LD.30.6
examples (icon) LDd.2.1
icon.library (2.0) 90DN.24
icon.library changes (2.0) 90DN.24
icon.library LD.30.2
Icon2C tool SWT.4
info files LD.30.3
tooltypes LD.30.5
workbench LD.30.1
IDCMPUPDATE
IDCMP (2.0) 90DN.29
IDCMP
disk LD.7.3
gadgets LD.7.3
in AppShell 90DN.28
introduction 89DN.1.2
intuition LD.7
keys LD.7.3
menus LD.7.3
modify LD.7.3
mouse LD.7.3
new classes (2.0) 90DN.29
prefs LD.7.3
requesters LD.6.1, LD.7.3
timing LD.7.3
verify LD.7.3
windows LD.7.3
ID
clipboard LD.32.4
IEEE
See also Math
articles AMT1.13
intro AMT1.13.1
math libraries LD.28.6
Iffparse.library
autodocs 90d.2.4
description 89DN.14
examples (looki.c) 90d.2.4, ND20.3.7
examples (preliminary 2.0) 89d.2.13
examples (sift.c) 90d.2.4, ND20.3.7
examples (simple.c) ND20.3.7
examples (tiff.c) 90d.2.4, ND20.3.7
examples (tiff2.c) ND20.3.7
examples (tree.c) 90d.2.4, ND20.3.7
examples (wiff.c) 90d.2.4, ND20.3.7
examples (wiff2.c) ND20.3.7
fd 90d.2.4
iffparse.doc ND20.3.7
iffparse.h ND20.3.7
iffparse.i ND20.3.7
iffparse.man ND20.3.7
iffparse_lib.fd ND20.3.7
includes 90d.2.4
library (preliminary 2.0) 89d.2.13
using 90DN.11
IFF
(c) IA.I
See also Iffparse.library, Interchange File Format
1988 news 88DN.5
1989 news 89DN.14
8SVX.CHAN spec IFF.5.7, IFFd.7
8SVX.PAN spec IFF.5.7, IFFd.7
8SVX AMT1.7.3, IFF.3.4
ACBM spec IFFd.7
ACBM IFF.5.1
ANBM spec IFFd.7
ANBM IFF.5.2
ANIM code 88d.2.2
ANIM spec IFFd.7
ANIMbrush 90DN.11
ANIM 88DN.3, IFF.5.3
ANNO IA.I
articles AMT1.7
ATAK IA.I
AUTH IA.I
BMHD IA.I
BODY AMT1.7.3, IA.I
CAMG (2.0) AMT1.7.5
CAMG IA.I
CAT AMT1.7.3, IA.I
CCRT IA.I
CHRS IA.I
chunk ids 89DN.14
clipboard data LD.32.3
clipboard 88DN.15, AMT1.14.2
CLUT AMT1.7.4
CMAP AMT1.7.4, IA.I
Compression AMT1.7.1
compression IA.I
CRNG IA.I

data chunk 89DN.14
deep ILBMs 89DN.14, AMT1.7.4
design theory IFF.4.4
DEST IA.I
disk ordering (CATS) AMT1.7.2
document ordering (CATS) AMT1.7.2
examples (8SVX player) AMT1.7.3
examples (ANIM) 88d.2.2, IFFd.6
examples (code modules) IFFd.2
examples (cycvb.c (vb int)) IFFd.6
examples (Display (1.2)) IFFd.6
examples (display mode id (2.0)) AMT1.7.5
examples (FTXT clipboard) LD.32.6
examples (iff code) IA.I
examples (iffparse.library) 89DN.14
examples (PGTB) 88d.2.2, IFFd.6
examples (ScreenSave (1.2)) IFFd.6
examples (ShowAnim) 88d.2.2, IFFd.6
examples (SMUS player) AMT1.7.2
FONS IA.I
FORM and Chunk registry IFF.4.2
FORM IA.I
FTXT IA.I, IFF.3.2
general spec IFF.2
GRAB IA.I
HEAD spec IFFd.7
HEAD IFF.5.4
iffparse.library 89DN.14, 90DN.11
ILBM intro AMT1.7.1
ILBM.DPPV spec IFFd.7
ILBM.DPPV IFF.5.5
ILBM AMT1.7.4, IA.I, IFF.3.1
include files (EA) IFFd.3
include files IFF.6.1
INS1 IA.I
INST IA.I
introduction IA.I
intro IFF.1
LIST AMT1.7.3, IA.I
manual IA.I
MCC-compat. obj modules (EA) IFFd.5
NAME IA.I
PGTB spec IFFd.7
PGTB 88DN.4, IFF.5.6
preferences (2.0) 90DN.10
property chunk 89DN.14
PROP IA.I
RLSE IA.I
SHDR IA.I
SMUS player AMT1.7.2
SMUS IA.I, IFF.3.3
sound files AMT1.6.6
source listings IFF.6.2
SPRT IA.I
standards 90DN.17
structure IA.I
supplementary docs IFFd.4
TRAK IA.I
VHDR IA.I
WORD spec IFFd.7
WORD IFF.5.8
Ignore
port (PA_IGNORE) LD.18.2
ILBM
See also IFF
24-bit AMT1.7.4
AboutILBM (summary) IFFd.4
bit ordering (deep) 89DN.14
CAMG format (2.0) 90DN.11, AMT1.7.5
clipboard AMT1.14.2
Display program (1.3) IFF.7.5, SWT.8
DPPV chunk spec IFF.5.5, IFFd.7
IFF spec IFF.3.1
introduction AMT1.7.1
intro IFF.4.3
ScreenSave program (1.3) IFF.7.5, SWT.8
ScreenSave program (2.0) 90d.1.2
Image
3D AMT1.14.3
boopsi 90DN.32
color LD.8.1
custom (2.0) 90DN.29
data LD.8.1
description LD.8.1
gadget LD.4.3
image structure LD.8.1
intuition LD.8
manipulation (CDTV) 90DN.12
position LD.8.1
vsprite LD.25.4
Include files
assembler (1.3) IA.E
C language (1.3) IA.D
electronic (1.3 stripped) ND13.3.7
electronic (1.3) ND13.2.1
electronic (2.0) ND20.2.1
filesysres.h 88d.4.4, 89d.2.3
hardblocks.h 88d.4.4, 89d.2.3
IFF (EA) IFFd.3
scsidisk.h 88d.4.4, 89d.2.3
INET
See also Network
TCP/IP 90DN.20
Information transfer
mouse style LD.11.5
Information
memory LD.15.3
Initialization
list LD.16.3
Input
articles AMT1.11
commodities 90DN.25
CON: window ADOS.2.5
console LD.33, LD.33.3
DOS file handle ADOS.2.2
examples (idio.c) 88d.1.8
examples (iefa.asm) 88d.1.8
examples (inputhandler.a (1.3)) LDd.2.16
examples (inputmake.c (1.3)) LDd.2.16
examples (inputset.c (1.3)) LDd.2.16
examples (inputswap.c (1.3)) LDd.2.16
examples (joymouse.c) 88d.1.8, AMT1.11.2
examples (joystick port output) AMT1.11.4
examples (keymapping) AMT1.11.3
examples (propjoy.c) 88d.1.8, AMT1.11.2
examples (swap input) LD.35.4
exec device LD.19
gameport.device LD.34
gameport AMT1.11.2
handler LD.35
input.device LD.35
intuition LD.7
joystick AMT1.11.2
keyboard AMT1.11.1, AMT1.11.3
low level allocation AMT1.11.2
resources AMT1.11.2
serial.device LD.40.5
trackdisk LD.42.8
INS1
See also IFF
in IFF AMT1.7.2
Integer
gadget LD.4.3
gadtools (2.0) 90DN.30
Interchange File Format
See also IFF
Interface
hardware HW.8
ports (hardware) HW.appxE
Interlace
See also Lace
copper (low-level) HW.2.8
display mode LD.23.1, LD.23.2
in CAMG AMT1.7.5
playfield (low-level) HW.3.2
screen viewmode LD.2.5
Internationalization
locale.library plans 90DN.26
International
compatibility articles AMT1.1
compatibility 89DN.21, LD.33.11
examples (compat.c) 89d.2.15
examples (strings.asm) 88d.1.5
keyboard compatibility AMT1.1.4
keyboard input AMT1.11.1
testing AMT1.1.10, AMT1.1.5
translation 90DN.13
travel AMT1.1.6
Interrupt
audio (hardware) HW.7.5
blitter (hardware) HW.6.10, HW.7.5
control register (hardware) HW.appxF.6
copper (hardware) HW.7.5
disabling LD.21.4
disk (hardware) HW.7.5
examples (assembler) 88d.1.4
examples (read buffer full) LDd.1.19
examples (softint) LDd.1.19
examples (vertical blank) LDd.1.19
external (hardware) HW.7.5
handler LD.21.2
hardware HW.7.5
master (hardware) HW.7.5
nonmaskable LD.21.1
priorities LD.21.1
processor LD.21
server LD.21.2
software AMT1.9.5, LD.21.3
vertical blank (hardware) HW.7.5
Intonation
speech LD.37.5
IntuiText
color LD.8.1
description LD.8.1
draw mode LD.8.1
font LD.8.1
intuitext structure LD.8.1
position LD.8.1
IntuitionBase
locking LD.10.2
Intuition
alerts LD.6.2
articles AMT1.4
boopsi (2.0) 90DN.32
boopsi (readme) ND20.3.3
boopsi classes (2.0) 90DN.29
boopsi 90DN.32
compatibility (2.0) 90DN.16
copper LD.10.5
custom gadget (readme) ND20.3.3
custom gadgets (1.4) 89DN.12.2
event processing AMT1.4.5
examples (2024.c (screens)) ND20.3.3
examples (alert (1.3)) LDd.1.8
examples (autorequest) LD.6.3
examples (autorequester.c (1.3)) LDd.1.8
examples (backdrop window) LD.3.3
examples (boopsi (2.0)) 90d.1.5
examples (boopsi demos) ND20.3.3
examples (borderless.c (1.3)) LDd.1.6
examples (buildreq) 88d.1.7
examples (cgmain.c (custom gadget)) ND20.3.3
examples (CloseWindowSafely) LD.7.4
examples (closewindowsafely.c (1.3)) LDd.1.11
examples (custgad (2.0)) 90d.1.5
examples (cwb.c (screens)) ND20.3.3
examples (dblmenureq.c (1.3)) LDd.1.8
examples (dcop) 88d.1.7
examples (dead key convert) LD.9.4
examples (democlasslib.c (boopsi)) ND20.3.3
examples (demoframe.c (boopsi)) ND20.3.3
examples (demoimage.c (boopsi)) ND20.3.3
examples (demopubi.c (boopsi)) ND20.3.3
examples (demotextb.c (boopsi)) ND20.3.3
examples (dialgadget.c (custom gadget)) ND20.3.3
examples (disabled render) AMT1.4.2
examples (display alert) LD.6.3
examples (displayalert.c (1.3)) LDd.1.8
examples (double-menu request) LD.6.3
examples (dual-playfield screen) LD.2.5
examples (dualpf.c (1.3)) LDd.1.7
examples (dvdemo) 88d.1.7
examples (EasyReq.c (2.0)) 90d.1.5
examples (ellipse.c (custom gadget)) ND20.3.3
examples (emboxclass.c (boopsi)) ND20.3.3
examples (emboxpubcl.c (boopsi)) ND20.3.3
examples (event loop) LD.1.4
examples (ez.c (easyrequest)) ND20.3.3

examples (font accomodation) 90d.3.15
examples (frame1class.c (boopsi)) ND20.3.3
examples (full screen) AMT1.4.1
examples (gadgets.c (1.3)) LDd.1.12
examples (gadgets.c) LD.4.5
examples (greetings.c (1.3)) LDd.1.7
examples (hookface.asm) ND20.3.3
examples (IDCMP event loop) LD.7.4
examples (idcmpdemo.c (1.3)) LDd.1.11
examples (ihooks (prelim 2.0)) 89d.1.3
examples (image data) LD.8.1
examples (images.c (custom gadget)) ND20.3.3
examples (invisible pointer) LD.3.3
examples (limit keystroke repeat) AMT1.4.4
examples (lines.c (1.3)) LDd.1.6
examples (lo-res screen) LD.2.5
examples (locklist.c (screens)) ND20.3.3
examples (memory allocation) LD.10.3
examples (menu) LD.5.2
examples (menus.c (1.3)) LDd.1.12
examples (modular event handling) AMT1.4.5
examples (mouse and keyboard) LD.9.4
examples (mousekeys.c (1.3)) LDd.1.10
examples (mutual exclude) AMT1.4.3
examples (myclass.library (boopsi)) ND20.3.3
examples (myclassinit.c (boopsi)) ND20.3.3
examples (myclasslib.asm (boopsi)) ND20.3.3
examples (mygroupclass.c (boopsi)) ND20.3.3
examples (mymodelclass.c (boopsi)) ND20.3.3
examples (open library) IA.A
examples (oscan) 88d.1.7
examples (overscan 1.3) 88d.1.7
examples (pointer to image) AMT1.5.1
examples (prop gadget) LD.4.3
examples (pub.c (screens)) ND20.3.3
examples (pubi.c (boopsi)) ND20.3.3
examples (pubsc.c (prelim 2.0)) 89d.1.3
examples (pubsc.c) ND20.3.3
examples (remembertest.c (1.3)) LDd.1.9
examples (renderinfo.c) 90d.3.15
examples (scdemo (2.0)) 90d.1.5
examples (scdemo.c (prelim 2.0)) 89d.1.3
examples (scdemo.c) ND20.3.3
examples (sdm.c (screens)) ND20.3.3
examples (strdemo (2.0)) 90d.1.5
examples (strgad.c) ND20.3.3
examples (strhooks.c) ND20.3.3
examples (superbitmap) LD.3.3
examples (tagdebug.c (booopsi)) ND20.3.3
examples (textbclass.c (boopsi)) ND20.3.3
examples (two window) LD.3.3
examples (twowindows.c (1.3)) LDd.1.6
examples (ucop) 88d.1.7
examples (visitor.c (screens)) ND20.3.3
examples (vm.c (screens)) ND20.3.3
font use 88DN.10
functions LD.12
gadgets AMT1.4.2, AMT1.4.3, LD.4
general (readme) ND20.3.3
graphics LD.8.1
IDCMP LD.6.1, LD.7
images LD.8
input and output LD.7
input.device LD.35.3
internal procedures LD.13
introduction (IDCMP) 89DN.1.2
introduction 89DN.1.2
intuition.library LD.1
IntuitionBase LD.10.2
intuition LD.11
keyboard LD.9.3
line drawing LD.8
manual LD.1
memory allocation LD.10.3
menus LD.5
mouse LD.9.1
overview (1.4) 89DN.12
overview (2.0) 90DN.29
pointer AMT1.5.1
Preferences LD.10.4
public screen (readme) ND20.3.3
requesters LD.6.1
screens LD.2
sprites LD.10.8
style LD.11
takeover 88DN.9
text LD.8
time (current) LD.10.6
view address LD.8.1
view port address LD.8.1
windows LD.3
INVERSVID
draw mode LD.23.4
IPrefs
preferences (2.0) 90DN.10
ISA
board form factor 90DN.2
Italic
text LD.24.6
ITEMNUM
menu LD.5.1
Item
menu LD.5.1
JAM1
draw mode LD.23.4
JAM2
draw mode LD.23.4
Janus
See also Bridgeboard
autodocs (1989) 89DN.7.2
janus.library 89DN.7
version 2.0 89DN.7
Joyport
examples (output) 88d.4.3
Joystick
absolute LD.34.2
controller port (hardware) HW.8.2
digital (hardware) HW.8.2
examples (joymouse.c) 88d.1.8
gameport.device LD.34.2
input AMT1.11.2
pin out (hardware) HW.appxE
port output AMT1.11.4
proportional (hardware) HW.8.2
relative LD.34.2
KCmpStr
debug.lib LD.appxB.3
Keyboard
compatibility AMT1.1.3
CON: input ADOS.2.5
console input LD.33, LD.33.3
dead key handling LD.9.4
diagram (hardware) HW.8.4
examples (keyhandler.a (1.3)) LDd.2.15
examples (keyinput.c (1.3)) LDd.2.15
examples (keymatrix.c (1.3)) LDd.2.15
examples (keyreset.c (1.3)) LDd.2.15
examples (matrix) LD.36.2
examples (read event) LD.36.3
examples (reset handler) LD.36.2
hardware HW.8.4, HW.appxH
input.device LD.35.1
input AMT1.11.1, AMT1.11.3
international AMT1.11.1
intuition LD.9.3
keyboard.device LD.36
keycodes LD.33.9
matrix (hardware) HW.appxH.9
mouse control LD.9.4
pin out (hardware) HW.appxE
RAW input ADOS.2.5
repeat limit (2.0) 90DN.29
repeat AMT1.4.4, LD.35.2
reset (hardware) HW.appxH.6
reset handler LD.36.2
Keymap
conversion LD.33.11
description LD.33.11
examples (example source) LD.33.12
examples (keymap.c) 88d.1.5
Keytoy tools SWT.9
national AMT1.11.1, AMT1.11.3
use of 88DN.16
Key
animation LD.25.10
KGetChar
debug.lib LD.appxB.3
KGetNum
debug.lib LD.appxB.3
KMayGetChar
debug.lib LD.appxB.3
KPrintF
debug.lib LD.appxB.3
KPutChar
debug.lib LD.appxB.3
KPutStr
debug.lib LD.appxB.3
Lace
See also Interlace
screen viewmode LD.2.5
Layers
compatibility (2.0) 90DN.16
examples (clip window) LD.26.5
examples (layer operations) LD.26.5
examples (layer.c (1.3)) LDd.2.5
introduction 89DN.1.2
layers.library LD.26.1
routines LD.26.2
types LD.26.3
Left amiga key
style LD.11.4
Legal
software protection 89DN.9
Library
See also Linker Lib
adding LD.14.3
arexx 89DN.11
autodocs (1.3) IA.A
cache pointers LD.14.2
calling (assembler) 88DN.20
calling LD.14.2
compatibility AMT1.1.9
creating 88DN.22, LD.14
creation AMT1.3.4
debugging tools 90d.3.5
dos.library ADOS.2.2
examples (library source) IA.G
examples (source) IA.G
function index (1.3) IA.J
introduction 89DN.1.1
layout LD.14
library structure LD.14.3
LVO LD.14.2
manual LD.Preface
required functions LD.14.3
sample library (1989) 89d.2.3
sample library (1990) LDd.2.6
set function LD.14.3
version list IA.A
wedging 88DN.22
Lib
See also Linker Lib
Light pen
port (hardware) HW.8.2
Light source
style LD.11.7
Limits
window LD.3.2
Line
blitter 88DN.11
draw LD.23.4
intuition LD.8
mode in blitter (hardware) HW.6.13
pattern LD.23.4
Linker libs
Amiga.lib 88d.4.3
amiga.lib LD.appxB.2
appshell.lib (preliminary 2.0) 90d.2.5
Astartup.obj 88d.3.3
autodocs (1.3) IA.F
ddebug.lib 90d.3.10, ND20.3.19
debug.lib 90d.3.10, LD.appxB.3, ND20.3.19
electronic (1.3) ND13.3.7
intro LD.appxB.1
Linker
Alink ADOS.2.4
overlays ADOS.2.4
tools (1988) SWT.10

WITH files ADOS.2.4
Lists
assembler macros LD.16.4
exec LD.16
file block ADOS.3.1
Initialization LD.16.3
interrupt LD.21.2
list structure LD.16.2
management LD.16.3
min list structure LD.16.2
min node structure LD.16.2
node structure LD.16.2
scanning LD.16.6
search LD.16.3
shared LD.16.7
Listview
gadtools (2.0) 90DN.30
LIST
iff definition IA.I
Loadseg
examples (loadseg.asm) 88d.3.4
Load
file layout ADOS.3.2
Locale.library
examples (catalogue.sample (2.0)) 90d.2.2
examples (CatGen (2.0)) 90d.2.2
examples (getmsg.c (2.0)) 90d.2.2
examples (locale_lib.asm (2.0)) 90d.2.2
plans 90DN.26
Locking
Novell record locking 90DN.21
Lock
audio LD.31.2
file system ADOS.3.3
file ADOS.2.2
handling (DOS (2.0)) 90DN.31
intuition base LD.10.2
layer LD.26.2
ShowLocks tool SWT.10
Low resolution
display mode LD.23.1, LD.23.2
LVO
electronic (1.3) ND13.3.7
exec library LD.14.2
generating from FD 90d.3.9
library vector offsets (1.3) IA.H
Macintosh
sound files AMT1.6.6
Manual
DOS ADOS.2
exec LD.14
graphics LD.23
hardware HW.1.1
intuition LD.1
libraries and devices LD.Preface
Manufacturer number
assignment AMT1.12.4
Marketing
export issues 89DN.26
international 88DN.35, 89DN.24
legal protection 89DN.9
public relations 88DN.19, 89DN.25
target markets 89DN.24
translations 90DN.13
trends 1988 88DN.30
Maskable
interrupt (hardware) HW.7.5
Mask
blitter (hardware) HW.6.5
blitter 88DN.11
bob imageshadow LD.25.7
collision AMT1.5.6, LD.25.8
graphics write mask LD.23.4
Master
A3000 local bus 90DN.1
MathIEEE.resource
information AMT1.13.1
Math
articles (IEEE) AMT1.13
examples (div0 IEEE) 88d.3.2
examples (div0.c (1.3 IEEE)) ND13.4.3
examples (fast fourier transforms) AMT1.13.2
examples (ffpconv.c (1.3)) LDd.1.4
examples (ffuz.c (1.3 IEEE)) ND13.4.3
Fast Fourier Transforms AMT1.13.2
FFP basic math LD.28.3
FFP conversion LD.28.5
FFP data format LD.28.2
FFP trancendental math LD.28.4
IEEE basic math LD.28.7
IEEE benchmark sources 88d.3.2
IEEE double precision 88DN.28
IEEE intro AMT1.13.1
IEEE libraries 88DN.28
IEEE trancendental math LD.28.8
libraries LD.28
math libraries (IEEE) LD.28.6
time arithmetic LD.41.7
Me mask
using AMT1.5.6
Mechanicals
86 Pin Slot Expansion Board TREF.appxA.7
A2000 Expansion Board Layout TREF.appxA.4
A2000 Form Factor TREF.appxA.5
A2000 Video Card TREF.appxA.6
A2000/B2000 Keyboard Connector Pinout TREF.appxA.8
A3000 local bus 90DN.1
A500 Exterior (86 pin Expansion Connector) TREF.appxA.3
A500/2000 Mouse Diagram and Pinout TREF.appxA.9
Zorro III card 90DN.2
MEmacs
user docs (1.3) EN13.7
Memory
allocation (exec) LD.15
allocation through intuition LD.10.3
amiga memory map 89DN.7.1
clear LD.23.4
compatibility AMT1.1.3, AMT1.1.9
copy LD.15.4
debugging tools (1988) SWT.7
debugging tools 90DN.12
free (automatic) LD.15.6
header structure LD.15.7
information LD.15.3
introduction 89DN.1.1
janus (dual-port) 89DN.7.1
multiple allocations LD.15.5
pc memory map 89DN.7.1
printing with low memory LD.39.3
problems AMT1.1.11, LD.appxA
Snoop tool SWT.10
system memory map (hardware) HW.appxD
type LD.15.2
viewport LD.23.2
Zorro memory map 90DN.2
Menu shortcut
style LD.11.4
MENUHELP
IDCMP (2.0) 90DN.29
MENUNUM
menu LD.5.1
Menu
about LD.5.1
appmenu description (2.0) 90DN.24
Checkmark LD.5.1
clear LD.5.1
command key LD.5.1
edit menu LD.11.1
enable/disable LD.5.1
examples (complete) LD.5.2
examples (menus.c) LDd.1.12
functions LD.5.1
gadtools (2.0) 90DN.30
intuition LD.5
item LD.5.1
layout (gadtools (2.0)) 90DN.30
menu structure LD.5.1
menuitem structure LD.5.1
mutual exclude LD.5.1
off (RMBTRAP) LD.5.1
project menu LD.11.1
set LD.5.1
shortcut LD.5.1
style LD.11.1
subitem LD.5.1
toggle LD.5.1
verify LD.5.1
window functions LD.3.2
workbench (2.0) 90DN.24
Message
arexx 89DN.11
between processes AMT1.2.12
exec LD.18.3
ignore LD.18.2
introduction 89DN.1.1
mouse LD.9.2
port LD.18.2
signal LD.18.2
software interrupt LD.18.2
wbstartup LD.30.4
MFM encoding
disk LD.appxC
Micro Hz
timer unit LD.41.2
Midi
ARexx command set (preliminary) AMT1.14.6
assembler output AMT1.9.6
examples (Midi output) AMT1.9.6
SMUS correspondence AMT1.7.2
time-stamping LD.41.1
Minterms
blitter (hardware) HW.6.4
Misc.resource
description LD.43.4
examples (flushdevice.c (1.3)) LDd.2.22
examples (serial and parallel) AMT1.9.1
using AMT1.9.1
MMU
debugging tools 90DN.12
Mode ID
in CAMG AMT1.7.5
Mode
display AMT1.5.5
text drawing LD.24.5
Modulation
audio hardware HW.5.3
Modulo
blitter (hardware) HW.6.3
definition (hardware) HW.3.5
definition AMT1.5.3
Monitors
capabilities AMT1.5.5
support (1.4) 89DN.6
types AMT1.5.5
MoreCache
function call AMT1.2.3
Motherboard
audio differences AMT1.6.4
revisions AMT1.14.4
Motion control
animation LD.25.10
Motor
control (trackdisk) AMT1.8.2
drive motor on/off LD.42.8
Mountlist
fast file system AMT1.2.1
Mouse
buttons in workbench (2.0) 90DN.24
control from keyboard LD.9.4
controller port (hardware) HW.8.2
controller type LD.34.2
examples (joymouse.c) 88d.1.8
gameport.device LD.34.2
information transfer LD.11.5
intuition LD.9.1
messages LD.9.2
movement repeat limit (2.0) 90DN.29
queue limits (1.4) 89DN.12.1
select LD.11.5
style LD.11.5
Mouth
speech LD.37.3
Move
sprite LD.25.2
Msgport
creation LD.18.2
introduction 89DN.1.1

msgport structure LD.18.2
naming 88DN.6, AMT1.1.2
software interrupt AMT1.9.5
Multimedia
AmigaVision 90DN.6
authoring hints 90DN.22
Multiple
multi-serial support LD.40.9
Multisync
ECS support (1.4) 89DN.4.1
Multitasking
See also Process, Task
arbitration AMT1.2.12
blitter use (hardware) HW.6.9
blitter LD.23.4
examples (multi-process) AMT1.2.12
forbidding LD.17.6
process creation (2.0) 90DN.31
processes AMT1.2.12
setfunction issues 88DN.22
task LD.17
troubleshooting AMT1.1.11
Mutual exclude
gadget AMT1.4.3, LD.4.3
gadtools (2.0) 90DN.30
menu LD.5.1
task (semaphore) LD.17.6
Narrator
examples (narrator.c (1.3)) LDd.2.14
examples (speech) LD.37.7
narrator.device LD.37
Negative response
requester LD.11.3
Network
cards 90DN.4
Novell 90DN.21
overview 90DN.19
SANA 90DN.5
standard 90DN.5
TCP/IP 90DN.20
New look
overview (2.0) 90DN.29
NewList
amiga.lib LD.appxB.2
NFS
See also Network
software 90DN.20
Node
list LD.16.3
NonMaskable
interrupt (hardware) HW.7.5
interrupt LD.21.1
Normal display rows
PAL AMT1.1.4
Normal
text LD.24.6
Notification
DOS (2.0) 90DN.31
in boopsi 90DN.32
preferences (1.4) 89DN.15
preferences (2.0) 90DN.10
Novell
See also Network
client software 90DN.21
NTSC
genlock interface (low level) AMT1.12.6
versus PAL AMT1.1.4
Numeric display
gadtools (2.0)s (2.0) 90DN.30
Object file
layout ADOS.3.2
Object oriented
boopsi 90DN.32
Object
animation LD.25.10
boopsi 90DN.32
file layout ADOS.3.2
workbench LD.30.1
OEM
See also VAR
programs 90DN.14
OOPS
boopsi 90DN.32
overview 90DN.32
Open
device LD.19.5
Operating system
See also 1.3, 1.4, 2.0
overview LD.Preface
Output
audio LD.31
CON: window ADOS.2.5
console LD.33.3
DOS file handle ADOS.2.2
exec device LD.19
intuition LD.7
parallel.device LD.38
printer.device LD.39
serial.device LD.40.4
speech LD.37
trackdisk LD.42.8
window LD.33
Overscan
displays 88DN.7
examples (intuition 1.3) 88d.1.7
more rows 88DN.7
overview (1.4) 89DN.6.2
viewing area LD.23.1
Packet
arexx 89DN.11
dos.library (1.2) AMT1.2.3
dos.library AMT1.2.9
examples (clicolors (assembler)) AMT1.2.10
examples (set raw mode) AMT1.2.9
handler design 88DN.23
types ADOS.3.3
use in DOS ADOS.3.3
PAL equations
A1000 A1000.1.1
A2000 TREF.3.1
A3000 local bus 90DN.1
autoconfig 90DN.3
CAS 90DN.3
memory timing 90DN.3
refresh counter 90DN.3
Zorro III board control 90DN.3
PAL machine
A2000 switch AMT1.1.10
audio AMT1.6.1
check at run-time 89DN.21
compatibility testing AMT1.1.5
compatibility 90DN.12
genlock interface (low level) AMT1.12.6
part number AMT1.1.4, AMT1.1.6
screen size AMT1.4.1
versus NTSC AMT1.1.4
Palette
gadtools (2.0) 90DN.30
PAN
See also IFF
Parallel
allocating hardware LD.43.4
articles AMT1.9
connector (hardware) TREF.1
debugging tools 90DN.12
examples (parallel.c (1.3)) LDd.2.13
examples (parallel2.c (1.3)) LDd.2.13
examples (parallel3.c (1.3)) LDd.2.13
hardware HW.8.5
parallel.device LD.38
pin out (hardware) HW.appxE
resource AMT1.9.1, LD.43.4
timing (hardware) HW.appxE
Parameters
parallel LD.38.4
Parity
serial LD.40.7
Parsing
command line (2.0) 90DN.31
iffparse.library 89DN.14, 90DN.11
Patent
software 89DN.9
Path
from C language AMT1.2.5
internals AMT1.2.3, AMT1.2.5
Pattern
draw patterns LD.23.4
matching (2.0) 90DN.31
Paula
See also Chip
pin assignment (hardware) HW.appxC
Pen
drawinfo specification (2.0) 90DN.29
drawing pens LD.23.4
Period
constants for audio AMT1.6.1
Peripheral
design 89DN.8
Permit
tasks LD.17.6
PGTB
See also IFF
IFF spec 88DN.4, IFF.5.6, IFFd.7
Phonemes
speech LD.29
table LD.37.7
Phonetic
narrator LD.37.3
Phone
touch tone sound AMT1.6.3
Piracy
disk 88DN.31
Pixel
printer LD.39.4
read LD.23.4
write LD.23.4
PlaneOnOff
image LD.8.1
PlanePick
gel system LD.25.7
image LD.8.1
Playfield
hardware (hardware) HW.3
low level programming HW.3
register summary (hardware) HW.3.7
vsprite color interaction LD.25.5
Pointer
disable AMT1.5.4
movement (gadget) LD.4.3
set position LD.35.2
style LD.11.7
window LD.3.1, LD.3.2
Port
A1000 interfaces A1000.1.2
creation LD.18.2
device proc AMT1.2.3
examples (joyport_output) 88d.4.3
hardware interfaces HW.appxE, TREF.1
name conflicts AMT1.1.2
signal LD.18.2
software interrupt AMT1.9.5, LD.18.2
wbstartup message LD.30.4
Position
layer LD.26.2
Positive response
requester LD.11.3
Postscript
intro AMT1.10.1
troubleshooting AMT1.10.5
with Unix 89DN.16
Post
clipboard AMT1.14.2, LD.32.4
Potgo.resource
description LD.43.5
examples (read9n5.c (1.3)) LDd.2.23
using AMT1.11.4
Power budget
A1000 A1000.2.4
A2000 TREF.7.2
A500 TREF.7.2
Preferences
extensible (2.0) 90DN.10
functions LD.10.4
overview (1.4) 89DN.15
overview (2.0) 90DN.10
preferences structure LD.10.4
printer AMT1.10.2
program (1.3) 88DN.29

Presets
  preferences (2.0) 90DN.10
Primitives
  graphics LD.23
Printer driver
  See also Printer
  examples (colormaster (1.3)) ND13.4.8
  examples (colormaster) 88d.3.1
  examples (colormaster2 (1.3)) ND13.4.8
  examples (colormaster2) 88d.3.1
  examples (epsonq (1.3)) ND13.4.8
  examples (epsonq) 88d.3.1
  examples (epsonx (1.3)) LDd.2.12, ND13.4.8
  examples (hp (1.3)) LDd.2.12, ND13.4.8
  examples (hp) 88d.3.1
  examples (okimate20 (1.3)) ND13.4.8
  examples (okimate20) 88d.3.1
  examples (xerox_4020 (1.3)) LDd.2.12, ND13.4.8
  examples (xerox_4024) 88d.3.1
Printer.device
  See also Printer driver, Printer
Printer
  See also Printer driver
  12 bit planes 88DN.17, AMT1.10.3
  articles AMT1.10
  aspect ratio 88DN.17, AMT1.10.3
  calling printer.device AMT1.10.3, AMT1.10.7, LD.39.3
  changes (1.3) 88DN.17
  control sequence LD.39.3
  Control tool SWT.4
  device changes (1.3) 88DN.18
  driver notes (1.3) AMT1.10.2
  driver sources 88d.3.1, LDd.2.12, ND13.4.8
  driver writing 88DN.21, AMT1.10.6, LD.39.4
  error codes 88DN.17
  examples (drivers) LD.39.5, LDd.2.12
  examples (epsonq) LD.39.9
  examples (epsonx) LD.39.9
  examples (graphic dump) LD.39.3
  examples (hp_laserjet) LD.39.9
  examples (simple print functions) AMT1.10.7
  examples (wdump.c (1.3)) LDd.2.12
  examples (xerox_4020) LD.39.9
  graphic driver AMT1.10.6
  graphic dumps LD.39.3
  large bit maps AMT1.10.3
  Postscript intro AMT1.10.1
  Postscript troubleshooting AMT1.10.5
  preferences AMT1.10.2
  printer.device changes (1.3) AMT1.10.4
  printer.device AMT1.10.3, AMT1.10.4, LD.39
  PrinterTest tool 90d.3.6, SWT.9
  programming 88DN.17
  serial hookup 88d.4.15, 89d.2.11
  simple text output LD.39.2
  strip printing 88DN.17, AMT1.10.3, LD.39.3
  supported (1.3) EN13.4
printf
  amiga.lib LD.appxB.2
Priority
  bob LD.25.7
  interrupt (hardware) HW.7.5
  interrupt LD.21.1
  sprites (hardware) HW.4.5
  task LD.17.2
Processor
  See also 68000, 68020, 68030, Compatibility
  exception LD.17.7
  interrupts LD.21
  trap LD.17.8
Process
  arbitration AMT1.2.12
  CreateProc() ADOS.2.2
  creation (2.0) 90DN.31
  creation AMT1.2.12
  multiple AMT1.2.12
  startup requirements ADOS.2.1
  structures ADOS.3.3
Productivity
  support (1.4) 89DN.6.1
Products
  new for 1988 88DN.1
  new for 1989 89DN.13
Programming
  articles AMT1.1
  assembler 88DN.20
  audio LD.31.2
  bad practices AMT1.1.1
  C language 88DN.2
  configuration compatibility AMT1.1.9
  guidelines 88d.4.15, 89d.2.11
  hardware HW.1.3
  intro C 88DN.37
  intro structures 88DN.34
  intro 88DN.34
  low level display HW.3
  low-level 88d.4.15, 89d.2.11
  modular AMT1.4.5
  practices AMT1.1.8
  quick start 89DN.1
  rules AMT1.1.1, IA.Preface, LD.Preface
  startup code LD.30.6
  troubleshooting AMT1.1.11, LD.appxA
  tutorial 89DN.1
  Workbench LD.30.6
Project
  workbench LD.30.1
PropInfo
  gadget LD.4.4
Proportional
  controller (hardware) HW.8.2
  controller notes AMT1.11.2
  font LD.24.11
  gadget LD.4.3
  modify gadget LD.4.4
PROP
  iff definition IA.I
  iffparse.library 90DN.11
PRT:
  See also Printer
  printer.device LD.39.2
Public
  memory type LD.15.2
  screens (2.0) 90DN.29
  screens ND20.3.3
  semaphore LD.20
Punctuation
  speech LD.37.5
putchar
  amiga.lib LD.appxB.2
Qualifiers
  keymap LD.33.11
Queue
  blitter LD.23.4
Ramlib
  compatibility (2.0) 90DN.16
Ramsey
  See also Chip
  A3000 architecture 90DN.36
RAM
  expanding A3000 AMT1.12.7
  Zorro III ram board 90DN.3
RangeRand
  amiga.lib LD.appxB.2
RasInfo
  usage LD.23.2
RASSIZE
  macro LD.23.2
Raster
  display LD.23.1
Rastport
  layer LD.26.3
RAW:
  control sequence ADOS.2.5
RAW
  console I/O LD.33.8
  DOS windows ADOS.2.5
  event stream AMT1.2.9
  key matrix LD.36.2
  keymapping AMT1.11.3
RDB
  See also Rigid disk block
  rigid disk block LD.27.3
Readmes
  boopsi ND20.3.3
  clipboard ND20.3.9
  custom gadget ND20.3.3
  electronic (1.2) ND13.4
  electronic (1.3) ND13.4
  gadtools ND20.3.6
  iffparse.doc ND20.3.7
  iffparse.man ND20.3.7
  intuition ND20.3.3
  public screen ND20.3.3
  tools disk ND20.3.9
Record locking
  DOS (2.0) 90DN.31
Recovery
  alert LD.6.2
Rectangle
  clipping LD.26.5
  copy LD.23.4
  drawing LD.23.4
  fill LD.23.4
  layer LD.26.2
  scroll LD.23.4
Redirection
  CLI ADOS.1.2
Reentrant
  code 88DN.2
Refresh
  DRAM 90DN.3
  gadget LD.4.3
  simple LD.26.1
  smart LD.26.1
  superbitmap LD.26.1
  window display LD.3.1
Regions
  creating LD.26.5
Register
  address list (hardware) HW.appxB
  alphabetical list (hardware) HW.appxA
  CIA list (hardware) HW.appxF.2
  color register summary (hardware) HW.3.8
  include file HW.appxJ
  playfield register summary (hardware) HW.3.7
  sprite color register (hardware) HW.4.12
  sprite register summary (hardware) HW.4.11
  usage in interrupt LD.21.2
RemTOF
  amiga.lib LD.appxB.2
Repeat
  keyboard AMT1.4.4
  keymap table LD.33.11
  period LD.35.2
  threshold LD.35.2
Reply
  message LD.18.3
  port LD.18.3
Report
  bug reports LD.Preface
Requester
  See also AsLlibrary
  Cancel tool SWT.6
  changes (2.0) 90DN.29
  custom bit-map LD.6.1
  double-menu LD.6.1
  easy request (2.0) 90DN.29
  example (EasyReq.c (2.0)) 90d.1.5
  examples (autorequester) LDd.1.8
  examples (dblmenureq) LDd.1.8
  functions LD.6.1
  gadgets in LD.6.1
  intuition LD.6.1
  menu LD.5.1
  position LD.6.1
  rendering LD.6.1
  requester structure LD.6.1
  style LD.11.3
  window LD.3.2
Request
  commands LD.19.4
  console io LD.33.4
  device io LD.19.2
  standard request structure LD.19.2
  timer LD.41.4
Reset
  examples (cold reboot) AMT1.3.3

handler (keyboard) LD.36.2
keyboard (hardware) HW.appxH.6
programmatic AMT1.3.3
system (hardware) HW.7.8
Resident
internals AMT1.2.3
reentrant code 88DN.2
structure LD.14.3
Resolution
playfield (low-level) HW.3.2
Resource
autodocs (1.3) IA.C
cia.resource LD.43.3
disk.resource LD.43.2
low level input AMT1.11.2
misc.resource AMT1.9.1, LD.43.4, LDd.2.22
potgo.resource LD.43.5, LDd.2.23
standard practices AMT1.1.8
Reverse path
font LD.24.11
Revision
bump revision LD.30.2
RF monitor
pin out (A1000) HW.appxE
RGB
deep ILBMs AMT1.7.4
Right amiga key
style LD.11.4
Rigid disk block
See also Hardblocks
definition LD.27.3
implementation 90DN.33
ROM Wack
serial debugger LD.22
Romtag
init time LD.27.3
structure LD.14.3
ROM
autoconfig 90DN.3
drivers (autoboot) LD.27.3
font LD.24.11
warning AMT1.1.1
Root
block (FFS) AMT1.2.4
block (structure) AMT1.2.4
block (trackdisk) AMT1.8.1
file block ADOS.3.1
SANA
network standard 90DN.5
overview 90DN.20
Satisfy
clipboard LD.32.4
Scale
audio (hardware) HW.5.6
Scan rate
table AMT1.5.5
Scan
directory (2.0) 90DN.31
list LD.16.6
Schematics
3.5 Disk Board A1000.appxA.3
5.25 Disk Board A1000.appxA.4
A1000 Daughterboard A1000.appxA.2
A1000 expansion ram A1000.appxA.6
A1000 expansion A1000.appxA.5
A1000 motherboard A1000.appxA.7
A2000 TREF.appxB.1
A500 TREF.appxB.3
B2000 TREF.appxB.2
Backplane Examples TREF.appxA.1
examples (plug in card) TREF.appxA.2
Zorro III memory card 90DN.3
SCRAM
expanding A3000 AMT1.12.7
ramsey 90DN.36
Screen
characteristics LD.2.4
coersion (2.0) 90DN.29
display modes (2.0) 90DN.29
examples (2024.c) ND20.3.3
examples (cwb.c) ND20.3.3
examples (dualpf) LDd.1.7
examples (greetings) LDd.1.7
examples (locklist.c) ND20.3.3
examples (pub.c) ND20.3.3
examples (public screen (2.0)) 90d.1.5
examples (pubsc.c) ND20.3.3
examples (scdemo (2.0)) 90d.1.5
examples (sdm.c) ND20.3.3
examples (visitor.c) ND20.3.3
examples (vm.c) ND20.3.3
font 88DN.10
full size AMT1.4.1
getscreendata (2.0) 90DN.29
intuition LD.2
managing display LD.2.5
newscreen structure LD.2.5
public (1.4) 89DN.12.1
public (2.0) 90DN.29
public screen (readme) ND20.3.3
screen structure LD.2.5
sprite LD.10.8
style LD.11.6
Unix (/dev/screen) 89DN.16
Scroller
gadtools (2.0) 90DN.30
Scroll
auto-scroll (2.0) 90DN.29
display clip (1.4) 89DN.6.2
playfield (low level) HW.3.6
raster LD.23.4
SCSI
examples (modesense.c) 90d.1.1, ND20.3.8
include files (1.3.1) 88d.4.4, 89d.2.3
programming 90DN.33
Security
network 90DN.20
Seek
iffparse.library 90DN.11
Seglist
fake AMT1.2.12
structure ADOS.3.3
Segment
lists (DOS) ADOS.3.3
Select box
gadget LD.4.3
Selection
gadget LD.4.3
Select
extended (workbench) LD.30.1
mouse style LD.11.5
Semaphore
public LD.20
signal LD.20
task LD.17.6
usage LD.20
SendIO
function description LD.19.3
Sequence
animation LD.25.10
expansion LD.27.2
Serial
allocating hardware LD.43.4
articles AMT1.9
connector (hardware) TREF.1
debugging tools 90DN.12
debugging LD.22, LD.appxB.3
electrical characteristics (hardware) HW.appxE
examples (serial.c (1.3)) LDd.2.19
examples (serial2.c (1.3)) LDd.2.19
hardware HW.8.6
high speed LD.40.5
interrupt handler LD.21.2
low level programming LD.40.10
low-level 89DN.22
Midi AMT1.9.6
multiple port serial.device LD.40.9
multiple ports AMT1.9.3
parameters for serial.device LD.40.7
pin out (hardware) HW.appxE
resource AMT1.9.1, LD.43.4
serial.device LD.40
shift register (hardware) HW.appxF.5
timing (hardware) HW.appxE
Server
arexx 89DN.11
interrupt LD.21.2
register usage LD.21.2
SetFileDate
function call AMT1.2.3
SetFunction
hooking into libraries 88DN.22
library LD.14.3
SetPointer
using AMT1.5.4
SetRawMode
function call AMT1.2.3
Shadow
drawinfo specification (2.0) 90DN.29
Shared
lists LD.16.7
serial LD.40.7
SHDR
See also IFF
in IFF AMT1.7.2
Shift
blitter (hardware) HW.6.5
blitter 88DN.11, AMT1.5.3
Shine
drawinfo specification (2.0) 90DN.29
Shortcut
menu LD.5.1
Shut-up
autoconfig 89DN.8
Signal
generation LD.17.5
port (PA_SIGNAL) LD.18.2
semaphore LD.20
task allocation LD.17.5
waiting for LD.17.5
Simple refresh
layer LD.26.1, LD.26.3
window type LD.3.1
Size
graphic pen LD.23.4
Slave
A3000 local bus 90DN.1
Slider
gadtools (2.0) 90DN.30
Smart refresh
layer LD.26.1, LD.26.3
window type LD.3.1
SMUS
See also IFF
IFF spec IFF.3.3
player AMT1.7.2
Software error
causes AMT1.1.11, LD.appxA
Software interrupt
port (PA_SOFTINT) LD.18.2
Software
See also Compatibility
debugging LD.appxA
interrupt LD.21.3
legal issues 89DN.9
translation of 90DN.13
Sort
bob LD.25.7
Sound
audio hardware HW.5
audio.device LD.31.2
Macintosh files AMT1.6.6
SpecialInfo
gadget LD.4.4
Speech
hints LD.37.5
mouth rendering LD.37.3
narrator.device LD.37
phonemes LD.29, LD.37.5
translator.library LD.29, LD.37.2
Spelling
phonetic LD.37.5
sprintf
amiga.lib LD.appxB.2
Sprite
animation LD.25.1
Assignment (vsprite) LD.25.5
attached (hardware) HW.4.8

collision detection (hardware) HW.7.3
convert to image AMT1.5.1
disabling DMA AMT1.5.4
display mode LD.23.2
DMA AMT1.5.4
examples (use sprite) LD.25.2
hardware HW.4
in windows LD.10.8
low level programming HW.4
moving (hardware) HW.4.4
multiple (hardware) HW.4.5
overlap (hardware) HW.4.7
pointer AMT1.5.1
position (hardware) HW.4.1
priority (hardware) HW.4.5, HW.7.2
re-use (hardware) HW.4.6
register summary (hardware) HW.4.11
reserving LD.25.5
screen viewmode LD.2.5
SetPointer AMT1.5.4
simple LD.25.2
window pointer LD.3.2
Stack
examples (get stack size) AMT1.2.11
size at runtime AMT1.2.11
size in icon LD.30.3
task LD.17.3
Standard request
structure LD.19.2
Standards
See also Style
Amiga 90DN.17
AppShell 90DN.28
boopsi classes 90DN.32
file format (IFF) IA.I
keymap LD.33.11
need for 90DN.17
preferences (2.0) 90DN.10
user interface 90DN.28
Star (*)
CLI current window ADOS.2.5
Startup-sequence
user docs (1.3) EN13.5
Startup
examples (reentrant code) 88DN.2
examples (startup code) 88d.3.3, LD.30.6
message AMT1.2.12
workbench startup drawer (2.0) 90DN.24
workbench LD.30.4
State
task LD.17.2
Stencil
blit LD.23.4
draw LD.23.4
Stereo
See also Audio
vision AMT1.14.3
Stopwatch
timer.device AMT1.9.7
Stream
iffparse.library 90DN.11
Stress
speech LD.37.5
String gadget
keymapping AMT1.11.3
StringInfo
gadget LD.4.4
String
gadget LD.4.3
gadtools (2.0) 90DN.30
keymap entries LD.33.11
Structure
definitions (1.3) IA.D, IA.E
offsets (1.3) ND13.3.7
offsets list IA.H
Style
See also Standards
command key LD.11.4
cursor key LD.11.4
gadget LD.11.2
help key LD.11.4
intuition LD.11
light source LD.11.7
menu shortcut LD.11.4
menu LD.11.1
mouse LD.11.5
requester LD.11.3
screen LD.11.6
text LD.24.6
window LD.11.6
SubItem
menu LD.5.1
SUBNUM
menu LD.5.1
Superbitmap
refresh layer LD.26.1, LD.26.3
window type LD.3.1
Superhires
ECS support (1.4) 89DN.4.1
support (1.4) 89DN.6.1
Synchronize
vsprite LD.25.5
Synchronous
device requests LD.19.5
System call
making (2.0) 90DN.31
System configuration
compatibility AMT1.1.8
System gadget
definition (1.3) LD.4.2
System software
overview LD.Preface
System
time LD.41.7
Tags
overview (2.0) 90DN.29, 90DN.30
Takeover
grabbing input AMT1.11.2
techniques 88DN.9
Task
creation LD.17.3
exec LD.17
free memory LD.15.6
introduction 89DN.1.1
math libraries (using) LD.28.1
priority LD.17.2
scheduling LD.17.2
stack LD.17.3
states LD.17.2
task structure LD.17.2
termination LD.17.4
waiting LD.17.2
TAS
See also Test and set
article AMT1.12.8
TCP/IP
See also Network
software 90DN.20
Template
CLI command ADOS.1.2
command line (2.0) 90DN.31
Test and set
information AMT1.12.8
Testing
compatibility AMT1.1.5
software LD.appxA.2
tools (1990) 90d.3.1
Text display
gadtools (2.0) 90DN.30
Text
color LD.24.4
draw mode LD.24.5
examples (fontparade) LDd.1.1
examples (showdefaultfont) LDd.1.1
examples (showdrawmodes) LDd.1.1
examples (showopenfont) LDd.1.1
font accomodation 90DN.37
graphics LD.24
IntuiText LD.8.1
intuition LD.8
layout 88DN.10
Tick
clock frequency AMT1.1.10
timer unit LD.41.2
TIGA
See also A2410
A2410 hires card 90DN.35
functions 89DN.10
introduction 89DN.10
Timer.device
See also 8520, CIA, Timer
Timer
See also 8520, CIA
articles AMT1.9
avoid "busy wait" AMT1.1.8
CIA chip (hardware) HW.appxF
cia.resource AMT1.9.2
compatibility (2.0) 90DN.16
device interrupts AMT1.9.5
drive timing AMT1.8.4
examples (8520 timing) AMT1.9.4
examples (getsystime.c (1.3)) LDd.2.18
examples (Midi output) AMT1.9.6
examples (software interrupt) AMT1.9.5
examples (stopwatch) AMT1.9.7
examples (timemath.c (1.3)) LDd.2.18
examples (timer) LD.41.8
examples (timersoftint.c (1.3)) LDd.1.19
examples (timertest.c (1.3)) LDd.2.18
examples (using cia timer) AMT1.9.2
hardware HW.appxF
low level access AMT1.9.2, AMT1.9.4
microhertz LD.41.3
software interrupts AMT1.9.5, LD.21.3
stopwatch AMT1.9.7
timer.device AMT1.9.5, AMT1.9.7, LD.41
vblank LD.41.3
Time
A2000 jumper AMT1.1.10
arithmetic LD.41.7
current time (intuition) LD.10.6
DateStamp() ADOS.2.2
international 90DN.26
low-level clock 89DN.22
system LD.41.7
Timing
See also 8520, CIA, Timer
floppy disk (hardware) HW.8.3
IDCMP LD.7.3
keyboard (hardware) HW.appxH.1
low-level 88d.4.15, 89d.2.11
parallel (hardware) HW.ap
serial (hardware) HW.appxE
Zorro III 90DN.2
Title
screen LD.3.2
window LD.3.2
TMS34010
See also A2410
A2410 hires card 90DN.35
Toggle
menu LD.5.1
Tools (1988)
Wack SWT.2
Tools (benchmarks)
calcpi 88d.3.2
float 88d.3.2
savage 88d.3.2
whetstone 88d.3.2
Tools (clipboard)
CF (clip file) ND20.3.9
test.ftxt.no ND20.3.9
test.ftxt ND20.3.9
test.ilbm ND20.3.9
Tools (debugger/disasm)
Wack1.0 90d.3.11
Wack 90d.3.11
Tools (debugging)
Complainer.par ND20.3.13
Complainer 90d.3.4, ND20.3.13
CpuTrap.par ND20.3.13
CpuTrap 90d.3.4, ND20.3.13
crash3 ND20.3.11
crash4 ND20.3.11
crash5 ND20.3.11
crash8 ND20.3.11
crasha ND20.3.11
DevMon 90d.3.6, ND20.3.15

EatCycles ND20.3.14
enforcer.doc ND20.3.13
Enforcer.par ND20.3.13
Enforcer 90d.3.4, ND20.3.13
IO_Torture 89d.2.1, 90d.3.4
Io_Torture ND20.3.13
Lawbreaker 90d.3.4
LawBreaker ND20.3.13
MemMung 89d.2.1, 90d.3.4, ND20.3.13, SWT.7
PrinterTest ND20.3.15
RomWack SWT.10
SRT (1.3) 88d.4.7, 89d.2.1, 90d.3.2
srt.text ND20.3.11
SRT ND20.3.11
TNT (1.3/2.0) 90d.3.2
TNT ND20.3.11
TStat 88d.4.16, 90d.3.5, ND20.3.14
Wack1.0 ND20.3.20
Wack ND20.3.20
WatchMem 89d.2.1, SWT.7
Wedge 90d.3.5, ND20.3.14, SWT.7
Tools (development)
68000 Cards Tool SWT.9
AutoDoc 90d.3.9
BumpRev 90d.3.9
LVO 90d.3.9
PrintA 90d.3.8, SWT.10
Read SWT.10
SetParallel SWT.10
StripA SWT.10
StripC SWT.10
Tools (disk)
DFC 88d.4.1
DiskEd 90d.3.8, ND20.3.17, SWT.10
DosList ND20.3.17
PrintA ND20.3.17
ShowLocks ND20.3.17
Tools (DOS)
AbsLoad SWT.10
Cancel SWT.6
DosList 88d.1.2, 90d.3.8
ShowLocks 89d.2.1, SWT.10
Tools (general)
Comp 88d.4.16
SaveMem 88d.4.16
SearchFile 88d.4.16
SearchMem 88d.4.16
Tools (graphics)
CharSet ND20.3.21
ReadPixel ND20.3.21
ShowGfxBase ND20.3.21
Tools (IFF)
Display IFFd.6, SWT.8
GrabAnim 88d.2.2, IFFd.6
IFFCheck IFFd.1
ILBM2Raw IFFd.1
ILBMDump IFFd.1
Raw2ILBM IFFd.1
Read8SVX IFFd.1
ScreenSave 2.0 90d.1.2
ScreenSave IFFd.6, SWT.8
ShowAnim 88d.2.2, IFFd.6
ZapIcon IFFd.1
Tools (intuigfx)
BlitLab 88d.4.1
CharSet 90d.3.12
FontMover 90d.3.15
Icon2C SWT.4
IconMerge SWT.6
MKS_Lens 90d.3.12
MoreRows SWT.10
Palette SWT.4
ReadPixel 90d.3.12
SetFont SWT.6
ShowGfxBase 90d.3.12
WinList 90d.3.12
Tools (intuition)
Autodoc ND20.3.18
BumpRev ND20.3.18
LVO ND20.3.18
MKS_Lens ND20.3.21
WinList ND20.3.21
Tools (keymap)
KeyToy1000 SWT.9
KeyToy2000 SWT.9
Tools (linker)
Alink SWT.10
Atom SWT.10
Tools (lister)
ConfigDev 90d.3.7, ND20.3.16
Config 88d.4.16, 90d.3.7, ND20.3.16
DevList 90d.3.7, ND20.3.16
DosList 90d.3.8
IntList 90d.3.7, ND20.3.16
LibList 90d.3.7, ND20.3.16
ModList 90d.3.7, ND20.3.16
ShowLocks 90d.3.8
TaskList 90d.3.7, ND20.3.16
Tools (math)
benchmarks (IEEE) 88d.3.2
Tools (memory)
AddMem SWT.10
Drip 90d.3.13, ND20.3.10, SWT.10
EatMem 90d.3.13, ND20.3.10
Flush 90d.3.13, ND20.3.10
Frags 90d.3.13, ND20.3.10, SWT.10
FreeMap SWT.9
MemList SWT.7
MemMon 90d.3.13, ND20.3.10
MemWall 90d.3.13, ND20.3.10
MergeMem SWT.4
owner.doc ND20.3.10
Owner 90d.3.13, ND20.3.10
Peek 90d.3.13, ND20.3.10
Poke 90d.3.13, ND20.3.10
SetTaskName 90d.3.13, ND20.3.10
SnoopStrip 90d.3.13, ND20.3.10
Snoop 89d.2.1, 90d.3.13, ND20.3.10, SWT.10
SpareMem 90d.3.13, ND20.3.10
Tools (performance)
PerfMon SWT.9
Tools (printer)
Control SWT.4
PrinterTest 88d.3.1, ND13.4.8, SWT.9
PrintFiles SWT.4
PrtControl 88d.3.1
PSPrint 90d.1.2
Tools (testing)
EatCycles 90d.3.5
EatMem 90d.3.13
PrinterTest 90d.3.6
Tools
See also Examples, Subject, Tools (subject)
clipboard LD.32.3
debugging (1988) SWT.1
debugging (1989) 89d.2.1
debugging (1990) 90d.3.1
debugging 90DN.12
testing 90d.3.1
Tooltypes
default LD.30.1, LD.30.3
find tool type LD.30.2
icon.library LD.30.3
match tool value LD.30.2
tool types array LD.30.5
workbench LD.30.1
Toolwindow
icon.library LD.30.3
Trackdisk
articles AMT1.8
checksums AMT1.8.1
commands LD.42.3
examples (low-level control) AMT1.8.2
examples (track copy) LD.42.14
examples (trackcopy.c (1.3)) LDd.2.17
layout AMT1.8.1
low level access AMT1.8.2, AMT1.8.3, AMT1.8.4
trackdisk.device LD.42
Track
floppy disk LD.42.2
Trade secret
software 89DN.9
Trademark
software 89DN.9
TRAK
See also IFF
in IFF AMT1.7.2
Trancendental
FFP math LD.28.4
IEEE math LD.28.8
Translation
of software 90DN.13
Translator
translator.library LD.29
Transputer
concepts 88DN.36
Trap
handler LD.17.8
processor LD.17.8
Trigger
gameport LD.34.2
Troubleshooting
autoconfig design (hardware) HW.appxG.1
software AMT1.1.11, LD.appxA
Type
of memory LD.15.3
Underline
text LD.24.6
Unix
See also Amix
Amix 89DN.5
User interface
standards 90DN.28
style LD.11
user's view 88DN.26
User
clip rectangle LD.26.5
copper lists 88DN.8, LD.23.5
survey results 88DN.26
Vanilla
keymapping AMT1.11.3
Varargs
boopsi 90DN.32
tag lists (2.0) 90DN.29, 90DN.30
VAR
See also OEM
programs 90DN.14
VBlank
timer unit LD.41.2
Vector base register
using AMT1.3.1
Vector
global (BCPL) ADOS.3.3
interrupt LD.21.2
Venn diagram
blitter (hardware) HW.6.4
Verify
IDCMP LD.7.3
menu LD.5.1, LD.7.3
requester LD.7.3
size LD.7.3
Version
of library IA.A
Vertical blank
interrupt (hardware) HW.7.5
Video slot
board form factor 90DN.2
description TREF.3.4
form factor TREF.appxA.6
Videodisk
AmigaVision 90DN.22
Video
beam position (hardware) HW.7.4
check PAL/NTSC 89DN.21
ECS control (1.4) 89DN.4.1
inverse LD.23.4
pin out (hardware) HW.appxE
Viewport
address from intuition LD.8.1
definition LD.23.2
display modes LD.23.2
examples (preliminary 2.0) 89d.1.1
limitations LD.23.2
mode specification (1.4) 89DN.6.1
modes (1.4) 89DN.6
modes LD.23.2
specification LD.23.2
View

address from intuition LD.8.1
definition LD.23.2
vsprites LD.25.5
Virus
article AMT1.14.1
disk 88DN.31
eradicating 88DN.32
Volume
audio (hardware) HW.5.7
audio hardware HW.5.2
audio LD.31.2
Vowel
selection LD.37.5
Vsprite
animation LD.25.4
vsprite structure LD.25.3
WaitIO
function description LD.19.3
Wait
audio samples AMT1.7.3
avoid "busy wait" AMT1.1.8
blitter LD.23.4
port LD.18.3
task LD.17.2, LD.17.5
Waveform
audio hardware HW.5.1
WBArg
See also Argument
WBStartup
structure LD.30.4
Wedge
hooking into libraries 88DN.22
Wildcard
DOS (2.0) 90DN.31
use ADOS.1.1
Window
appwindow description (2.0) 90DN.24
CLI (window pointer) AMT1.2.13
clipping (layer) LD.26.5
CON: output ADOS.2.5
console bounds LD.33.8
console input LD.33
console output LD.33
current (DOS) ADOS.1.1
examples (backdrop window) LD.3.3
examples (borderless) LDd.1.6
examples (superbitmap window) LD.3.3
examples (superbitmap) LDd.1.6
examples (two windows) LDd.1.6
full size AMT1.4.1
general LD.3.1
intuition LD.3
limits LD.3.2
maximum size LD.11.6
newwindow structure LD.3.2
open without IDCMP LD.7.3
sprite LD.10.8
style LD.11.6
title LD.3.2
window functions LD.3.2
window structure LD.3.2
workbench 90DN.24
WORD
See also IFF
IFF spec IFF.5.8, IFFd.7
Workbench
articles AMT1.4
compatibility AMT1.1.3
environment LD.30.4
examples (appicon.c (2.0)) 90d.1.3
examples (appicon.c (prelim 2.0)) 89d.2.2
examples (appicon.c) ND20.3.1
examples (appicon.image) ND20.3.1
examples (appmenu.c (2.0)) 90d.1.3
examples (appmenu.c) ND20.3.1
examples (appmenuitem.c (prelim 2.0)) 89d.2.2
examples (appwindow.c (2.0)) 90d.1.3
examples (appwindow.c (prelim 2.0)) 89d.2.2
examples (appwindow.c) ND20.3.1
examples (CLI and WB startup code (1.3)) LDd.2.2
examples (icon) LDd.2.1
examples (icon_example.c (1.3)) LDd.2.3
examples (prargs.c (1.3)) LDd.2.3
examples (rkm_icon_example.c (1.3)) LDd.2.3
examples (startup code) LD.30.6
introduction 89DN.1.2
managing screen LD.2.5
overview (1.4) 89DN.3
overview (2.0) 90DN.24
program environment ADOS.2.1
related functions LD.30
screen LD.2.2
user docs (1.3) EN13.1
X Window
with Unix 89DN.16
X-on
See also Serial
serial LD.40.7
Xicon
execute script files AMT1.1.7
Zorro III
See also Expansion
board form factor 90DN.2
bus issues AMT1.12.8
compatibility (2.0) 90DN.16
design 90DN.3
spec 90DN.2
Zorro II
See also Expansion
bus issues AMT1.12.8
compatibility 90DN.2

Amiga Mail
Volume II
September/October 1990

From the Desk of Jeff Scherb

The other day I attended a sales meeting for Commodore's Consumer Sales division. Overall, the meeting was upbeat, since everyone believes we're going to have a great Christmas selling season with the A500 in the consumer channel. If there was one recurring theme to the meeting, it had to be this - differentiating the Amiga from everything else on the market is the best way to sell the machine. Commodore and its developers all need to work together to make this successful.

I sat in the sales meeting for two days with people who are working long and hard to sell the Amiga. And how are they doing it? By differentiating the Amiga from everything else on the market. By setting up demos so the buyer can see the difference. By training salespeople to use key Amiga applications. By planning a multi-million dollar ad campaign.

One of the most exciting parts of the meeting was the preview of the fall ad campaign. The basic theme is - "What do you do after your kid's brain outgrows Nintendo?". The ads don't put the Nintendo down, but instead ask the above question and also point out the things that the Amiga does that a simple game box cannot. My favorite TV spot preview goes like this: "Preparing your kid for college? There are scholarships for art (shot of paint program), design (shot of CAD program), writing (word processor)," and so on, until we get to a shot of a kid playing a video game, over which the announcer sadly says, "But there are no scholarships for Super Mario Brothers." This multi-million dollar campaign includes TV, newspapers, and magazines (like *People*). The ads specifically focus on differentiating the Amiga from the other choices the consumer can make.

I left the meeting feeling that the Amiga would finally achieve its long term success, but another thought cast a shadow over my good feeling. I started to think about a disturbing industry trend that could seriously damage our hopes for the long term success of the Amiga. I'm talking about protecting markets through litigation rather than innovation. The industry giants are bashing each other in the courts over "look and feel." Microsoft, Apple, and Xerox are battling over the appearance of a window on the screen. Lotus is suing Borland over the menu structures in the Quattro spreadsheet. These giants may have the resources to fight these battles, but most Amiga developers don't. Lotus has already put Paperback Software out of business over their spreadsheet.

This trend seems to have infected some Amiga developers. It has been reported to me that the lawyers of one of the larger Amiga developers have been sending letters to one of the newer developers threatening action over the "look and feel" of a particular screen gadget. Elsewhere, two Amiga publishers are countersuing each other over the distribution rights to a certain program. Strangely enough, these same two companies had reportedly been in cooperation with each other where both were in a position to profit.

None of the Amiga developers are big enough to waste money on things like this. If litigation becomes a trend among Amiga developers, everyone will suffer. The Amiga is already behind the other platforms in availability of applications. No Amiga developers will benefit from using the legal system to divide up the same pie among fewer developers. In the long run, there may be no pie left to divide. The only road to success with the Amiga is for all of us to help expand the market. Then we all get to divide up a bigger pie.

Differentiate and innovate, don't litigate. Help us show the world that the Amiga is different and much better. In the professional channel, multimedia is just taking off. The consumer channel is ready for an alternative with more socially redeeming value than a simple game box.

Let's not miss the boat because we are sitting on the dock arguing about which of us has a valid ticket. There's plenty of room for everyone. All aboard!

-JS